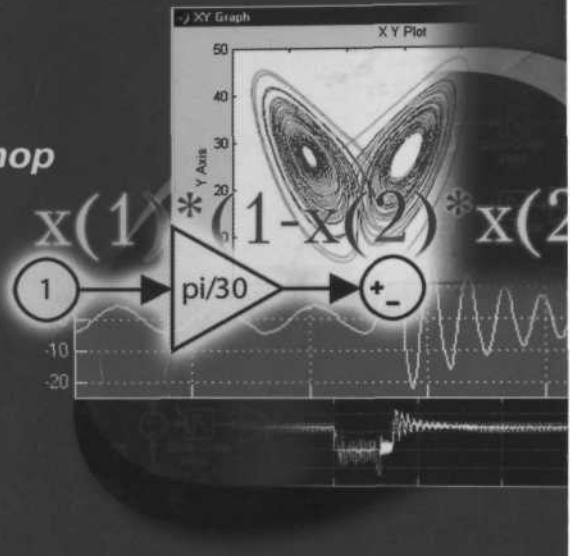
Нужная информация всегда под рукой!

Владимир Дьяконов

- *Nonlinear Control Systems*
- *Digital Signal Processor*
- *Power System*
- *Stateflow*
- *Real-Time Workshop*



SIMULINK 4

Специальный справочник

ПИТЕР®

Владимир Дьяконов

SIMULINK 4

Специальный справочник

Санкт-Петербург
Москва • Харьков • Минск
2002

ПИТЕР®

Владимир Дьяконов

Simulink 4. Специальный справочник

Главный редактор	*Е Строганова*
Заведующий редакцией	*И Корнеев*
Научный редактор	*Е Бочкарева*
Художник	*Н Биржаков*
Корректоры	*С Беляева, М Одинокова*
Верстка	*А Келле-Пелле*

ББК 32.973.23я22
УДК 681.3.068(03)

Дьяконов В.

Д93 Simulink 4. Специальный справочник. — СПб: Питер, 2002 — 528 с.: ил.

ISBN 5-318-00551-9

В новой книге профессора В Дьяконова впервые в нашей литературе обстоятельно рассмотрен один из основных пакетов расширения системы MATLAB 6 0 — Simulink 4 0 и его дополнительные компоненты. Совместно они образуют среду визуального имитационного и событийно управляемого моделирования с обширными инструментальными возможностями и богатейшими библиотеками блоков Книга весьма полно и с множеством практических примеров описывает как систему Simulink 4 0, так и ее окружение Дан также краткий курс по базовой системе MATLAB 6 0 Книга может служить руководством пользователя и справочным руководством по системе Simulink 4 0 и ее расширениям.

ISBN 5-318-00551-9

ЗАО «Питер Бук», 196105, Санкт-Петербург, Благодатная ул , д 67
Лицензия ИД № 01940 от 05 06 00
Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 95 3000 — книги и брошюры
Подписано к печати 19 10 01. Формат 60×90/16 Усл п л 33 Тираж 5000 Заказ 432

Отпечатано с готовых диапозитивов
в ФГУП ордена Трудового Красного Знамени «Техническая книга»
Министерства Российской Федерации по делам печати,
телерадиовещания и средств массовых коммуникаций
198005, Санкт-Петербург, Измайловский пр., 29

Краткое содержание

Содержание

Введение

Система MATLAB (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 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.

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. Прежде всего это достигается созданием целого ряда пакетов расширения системы, которые охватывают 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 полный доступ пользователям 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 и создания средств моделирования различных устройств и систем. В том числе удобных и очень наглядных визуально-ориентированных средств анализа, идентификации, построения и моделирования систем.

Новейшая версия системы MATLAB 6.0 поставляется вместе с пакетом расширения Simulink 4.0, предназначенным для моделирования динамических систем, модели которых составляются из отдельных блоков (компонентов). Этот пакет является самым ярким представителем программ, созданных на основе системы MATLAB. В нем реализованы принципы визуально-ориентированного программирования, что позволяет легко набирать нужные блоки и со-

единять их с целью составления модели системы или устройства. При этом сложнейшие уравнения состояния, описывающие работу моделей систем или устройств, 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 его надежности и достоверности Simulink 4.0 выгодно отличается от множества подобных программ. Особенно это относится к открытости пакета и возможностям пополнения его библиотек.

Вместе с базовой системой MATLAB 6.0, имеющей самые совершенные алгоритмы матричных вычислений и наиболее приспособленной для решения задач моделирования, Simulink 4.0 становится мощнейшим инструментом познания реалий мира путем их моделирования. И эти возможности многократно усиливаются десятками пакетов расширения системы MATLAB 6.0 + Simulink 4.0. Наиболее важные из них, которые разработчик систем корпорация MathWorks относит к разряду пакетов расширения Simulink 4.0, также описаны в этой книге. При этом важно подчеркнуть, что все эти пакеты представлены их новыми версиями, подчас существенно обновленными и расширенными.

В июне 2001 года фирма MathWorks объявила о начале поставок обновленной версии системы MATLAB 6.1 + Simulink 4.1. В данной версии, любезно предоставленной автору вскоре после этого объявления, имеется ряд усовершенствований. Например, в систему MATLAB включены три новые функции (одна относится к работе со строками, две другие — к работе со звуком). Пакет Simulink 4.1 пополнился несколькими новыми библиотечными блоками. Однако все эти изменения в малой степени касаются тех средств, которые описаны в данной книге. Поэтому ею могут пользоваться и читатели, имеющие возможность работать с комплексом MATLAB 6.1 + + Simulink 4.1. Его новые возможности детально описаны в соответствующем разделе справки по системе MATLAB 6.1, посвященном новинкам этой системы.

Состав книги

Как и в упомянутых ранее книгах, одной из главных трудностей при подготовке данной книги был отбор и систематизация огромного по

объему материала по системе MATLAB 6.0 и пакетам ее расширения. Некоторые из описанных в данной книге пакетов расширения имеют обширные описания во многие сотни страниц, представленные в справочной системе в виде HTML- и PDF-файлов. Поэтому естественно, что данная книга не претендует на полную и формальную замену этой весьма обширной и труднообозримой англоязычной документации.

В книге впервые в нашей литературе рассмотрены следующие средства новейшей системы MATLAB 6.0:

- базовая система MATLAB 6.0 (краткий курс);
- пакет моделирования динамических систем Simulink 4.0;
- пакет проектирования нелинейных систем Nonlinear Control Systems;
- пакет моделирования цифровых систем обработки сигналов Digital Signal Processor;
- пакет разработки электротехнических и энергетических систем Power System;
- пакет событийного моделирования систем Stateflow;
- мастерская реального времени Real-Time Workshop (обзорно).

Нетрудно заметить, что материал книги ориентирован на изучение не столько базовой версии системы MATLAB 6.0, сколько пакета расширения Simulink 4.0 и входящего в него программного окружения. Тем не менее, чтобы сделать книгу вполне самостоятельной, в нее включен краткий курс по базовой системе MATLAB 6.0 (глава 1).

Однако это описание не претендует на полноту. Достаточно полное описание базовой системы MATLAB 6.0 дается в учебном курсе автора по этой системе (см. главу 1, в начале которой дан список литературы). Читатели могут также воспользоваться и книгами по предшествующей версии системы, что (вместе с главой 1) позволит достаточно глубоко ознакомиться с возможностями системы MATLAB.

Краткое знакомство с пакетом расширения Simulink 4.0 дано в главе 2. Вместе с главой 1 они образуют краткий ознакомительный курс по системе MATLAB 6.0 + Simulink. В таком курсе нуждаются многие высшие учебные заведения, где рамки учебного плана не позволяют ввести детальное знакомство с этой мощной и, увы, достаточно громоздкой системой.

Главы 3–9 представляют собой базовый курс по системе имитационного моделирования динамических систем Simulink 4.0. Здесь достаточно полно описан интерфейс системы, набор ее библиотечных модулей и основы визуального моделирования разнообразных устройств и систем. Большое внимание уделено разбору кратких и простых примеров применения практически всех библиотечных блоков и рассмотрению более сложных демонстрационных примеров, включенных в поставку системы Simulink.

Главы 10–16 вместе с предшествующими главами образуют подробное справочное руководство по системе Simulink и ее программному расширению. В главе 10 описаны возможности Simulik 4.0 по подготовке своих библиотечных модулей и даже своих библиотек пользователями, то есть развитые средства расширения системы Simulink и ее ориентации на решение проблемно-ориентированных задач пользователя. Инструментальные средства системы Simulink 4.0 описаны в главе 11.

Начиная с главы 12 впервые в нашей литературе описаны новые версии пакетов расширения системы Simulink 4.0, которые разработчик системы фирма MathWorks относит к таковым. Глава 12 посвящена небольшому, но важному пакету проектирования нелинейных систем Nonlinear Control Systems. Интересной возможностью этого пакета является оптимизация нелинейных систем по различным критериям и возможность коррекции их параметров с целью получения заданных характеристик систем.

Глава 13 описывает мало известный нашим читателям пакет расширения Fixed-Point Blockset, предназначенный для моделирования систем с данными, заданными в формате с фиксированной точкой. Такие данные современные персональные компьютеры обрабатывают на аппаратном уровне предельно быстро. Это позволяет создавать модели устройств и систем с малыми затратами времени на моделирование. Такие модели лучше других приспособлены для последующей реализации на аппаратном уровне.

В главе 14 также впервые в нашей литературе довольно полно описан мощный пакет расширения Digital Signal Processor Blockset. Этот пакет ориентирован на разработку цифровых устройств обработки и фильтрации сигналов и является существенным расширением как библиотеки, так и возможностей системы Simulink. Он дает превосходные средства для моделирования дискретных систем самого различного назначения.

Глава 15 описывает новейшую реализацию пакета Power System Blockset, предназначенного 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: 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.

Наконец, глава 16 посвящена принципиально иному по назначению пакету моделирования систем — Stateflow Blockset. Здесь также описана новейшая реализация этого пакета. Базируясь на теории конечных автоматов, пакет обеспечивает событийное (ситуационное) моделирование систем с построением наглядных SF-диаграмм с элементами анимации, наглядно показывающими состояния отдельных объектов в ходе моделирования и связи между ними, меняемые при изменении событий. Этот пакет существенно дополняет возможности пакета Simulink и других пакетов расширения системы MATLAB.

Эта 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.

Поскольку книга посвящена конкретным и новейшим версиям программного продукта MATLAB 6.0 + Simulink 4.0, то при ее подготовке использовалась в основном фирменная документация (главным образом электронная). Поэтому список литературы в конце книги ограничен только работами (в основном книгами) отечественных авторов по предшествующим версиям системы MATLAB с ее расширениями. Необходимые сведения теоретического характера читатель найдет в многочисленных учебниках, справочниках и энциклопедиях по математике.

Графическая часть материала книги, включая схемы моделей и результаты моделирования, представлена в виде копий экрана с реально опробованных примеров. Это призвано создать у читателя впечатление от реального просмотра примеров и способствует существенному уменьшению ошибок в представлении примеров мо-

делирования систем. Тем не менее автор настоятельно рекомендуют сочетать чтение этой книги с реальной работой с описанными в ней пакетами. Иначе их изучение очень напоминает попытки обучения плаванию без воды или даже парашютным прыжкам без парашюта.

Предупреждения

Книги по современной 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. Автор считает нужным предупредить читателей об этом, хотя он сам 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 (особенно Simulink) требует от читателя знания основ математики, а применительно к данной книге — и основ моделирования, системотехники и системного анализа. Без этого невозможно гарантировать правильное применение используемых в системе методов и корректность получаемых результатов.

К сожалению, ограниченный объем книги и ее направленность исключили возможность рассмотрения важных, но весьма специфических и сложных вопросов работы системы MATLAB + Simulink с внешним оборудованием (за исключением, разумеется, дисковых накопителей и устройств печати). Речь идет о внешних аппаратных средствах, работающих под управлением системы MATLAB + Simulink и, в свою очередь, обеспечивающих управление этой системой. К числу таких программно-управляемых комплексов относятся промышленные роботы, измерительные комплексы, физические лаборатории и даже целые производства. Для их создания компьютеры оснащаются специальными контроллерами.

Такие системы в книге упомянуты лишь обзорно. Так же как и средства создания исполняемых модулей, такие как компилятор системы MATLAB, ускоритель работы Simulink и многочисленные пакеты расширения системы MATLAB.

Мы вынуждены предупредить читателя, что автор и издательство не несут никакой ответственности за ошибки пользователей (особенно учащихся) в освоении системы и за моральный или даже эко-

номический 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 6.0 + Simulink 4.0 заведомо имеются серьезные ошибки и недочеты. Просто такое предупреждение отвечает существующим нормам юридического права в отношении сложных программных продуктов и сопровождающей их документации.

Благодарности

Эта книга написана в инициативном порядке на кафедре физической и информационной электроники Смоленского государственного педагогического университета (СГПУ). В ней частично отражены материалы работ автора по гранту Министерства общего и профессионального образования РФ (применения современных систем компьютерной математики в решении фундаментальных задач естествознания) и по грантам Международной соросовской программы образования в области точных наук (ISSEP, гранты Соросовского профессора в области математики 1999 и 2001 гг.).

Автор благодарит представителя фирмы MathWorks (создавшей систему MATLAB) Наоми Фернандес (Naomy Fernandes). Стефана Вольфрама (S. Wolfram) — основателя и руководителя корпорации Wolfram Research Inc., создавшей и выпускающей мощные системы компьютерной математики класса Mathematica 4, автор благодарит за предоставленную возможность длительной стажировки в США на этой фирме. Это позволило не только набрать обширный материал по всем современным системам компьютерной математики, но и более четко определить место каждой из них. Что учитывалось при написании этой книги.

Генерального директора корпорации SoftLine (Россия) Игоря Боровикова и сотрудника этой фирмы Бориса Манзона за внимание к работе автора, поддержку и предоставление лицензии MathWorks и легальных версий системы MATLAB с ее расширениями для подготовки данной книги и других книг по этой системе.

Автор благодарит 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. Своим коллегам профессору В. Круглову и доцентам И. Абраменковой и А. Пенькову автор благодарен за предоставление и обсуждение отдельных материалов и примеров.

Адреса для переписки

С автором данной книги можно связаться по электронной почте: dyak@keytown.com. Автор заранее выражает признательность всем читателям, которые готовы сообщить свое мнение о данной книге. Кроме электронной почты автора замечания можно направлять по адресу: 214000, Смоленск, ул. Пржевальского 4, СГПУ.

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

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

Подробную информацию о наших книгах вы найдете на web-сайте издательства http://www.piter.com.

Глава 1. Знакомство с системой MATLAB 6.0

- Система MATLAB + Simulink (реализация 12)
- Начало работы с системой MATLAB
- Основные объекты MATLAB
- Операции с векторами и матрицами
- Операции сессии
- Работа с файлами
- Работа со средствами графики
- Специальные средства графики

Система MATLAB + Simulink (реализация 12)

Состав системы

Новая версия системы MATLAB 6.0, известная также как реализация 12, появилась в конце 2000 года. Она содержит мощную базовую систему MATLAB — матричную лабораторию и десятки пакетов расширения для самых разных областей компьютерной математики. Система основана на современных высокоэффективных алгоритмах матричных операций, которые базируются на известных и хорошо апробированных пакетах матричных вычислений LINPACK и EISPACK.

Все пакеты расширения системы MATLAB предполагают работу в ее среде. На рис. 1.1 показан состав системы MATLAB + Simulink + + пакеты расширения.

Хотя пакет моделирования динамических систем Simulink 4.0 имеет большое самостоятельное значение, его работа невозможна без системы MATLAB 6.0. Поэтому, чтобы сделать данную книгу полностью независимой, первую главу мы начнем с краткого описания

основ работы с MATLAB. Это описание можно рассматривать как мини-курс по MATLAB, который часто дается на спецкурсах в различных вузах.

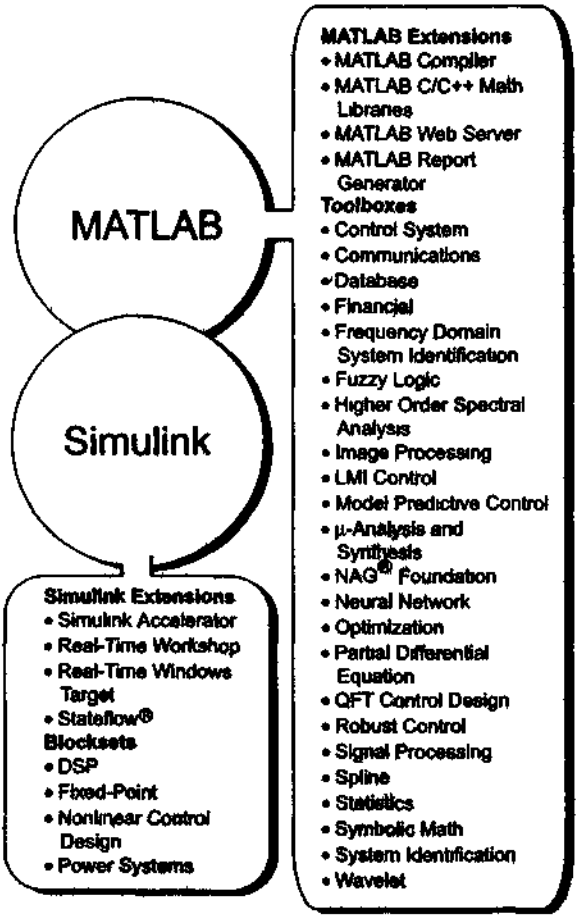


Рис. 1.1. Состав системы MATLAB+Simulink+пакеты расширения

Документация и литература по системе MATLAB

Фирменная документация по системе MATLAB и пакетам ее расширений содержит десятки тысяч страниц. Только описание MATLAB занимает около 5000 страниц в формате PDF, а полное описа-

ние системы в этом формате занимает целый компакт-диск. Оно дано на английском и японском языках.

В связи с этим детальное знакомство с системой MATLAB целесообразно с помощью приведенных ниже книг:

- Дьяконов В. П. MATLAB: учебный курс. СПб.: Питер, 2001.
- Дьяконов В. П. MATLAB 6: учебный курс. СПб.: Питер, 2001.
- Дьяконов В. П., Абраменкова И. В. MATLAB 5. Система символьной математики. М.: Нолидж, 1999.
- Дьяконов В. П., Абраменкова И. В., Круглов В. В. MATLAB 5 с пакетами расширений. М.: Нолидж, 2001.

Далее для простоты под системой MATLAB мы будем подразумевать систему MATLAB 6.0 (R12). Относящиеся к MATLAB термины «система» и «пакет» будем считать равноценными.

MATLAB — матричная лаборатория

Система MATLAB выполняет операции над массивами. Одномерный массив называют *вектором*, а двумерный — *матрицей*:

[1 2 3 4] или [1, 2, 3, 4]

Векторы из 4 элементов

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 8 & 7 & 6 \end{bmatrix}$$

Матрица размером 3 × 4

$$\begin{bmatrix} a & a+b & a+b/c \\ x & y*x & z \\ 1 & 2 & 3 \end{bmatrix}$$

Матрица с элементами разного типа

Массивы в общем случае характеризуются размерностью и размером. *Размерность* массива определяет его структурную организацию: в виде одной строки или одного столбца (размерность 1), страницы (размерность 2), куба (размерность 3) и т. д. MATLAB допускает задание и использование многомерных массивов ряда типов, в том числе массивов ячеек и записей.

Размер вектора — это число его элементов, а размер матрицы определяется числом ее строк m и столбцов n. Обычно размер матрицы указывают как $m \times n$. Матрица называется квадратной, если $m = n$.

Элементы векторов и матриц рассматриваются как *индексированные переменные*. Например, V_2 — второй элемент вектора V; $M_{2,3}$ — третий элемент второй строки матрицы M.

Интересно отметить, что даже обычные числа и переменные в MATLAB рассматриваются как матрицы размером 1×1, что дает единообразные формы и методы проведения операций над обычными числами и массивами Это также означает, что большинство вычислительных функций может работать с аргументами в виде векторов и матриц, вычисляя значения для каждого их элемента

Начало работы с системой MATLAB

Запуск MATLAB и работа в режиме диалога

В этой книге предполагается, что MATLAB используется в операционной системе Windows 95 OSR 2, Windows 98 или Windows 98 OSR 2 Рисунок 1 2 иллюстрирует запуск системы MATLAB 6 0 из главного меню операционной системы Windows 98 со стандартным видом Рабочего стола

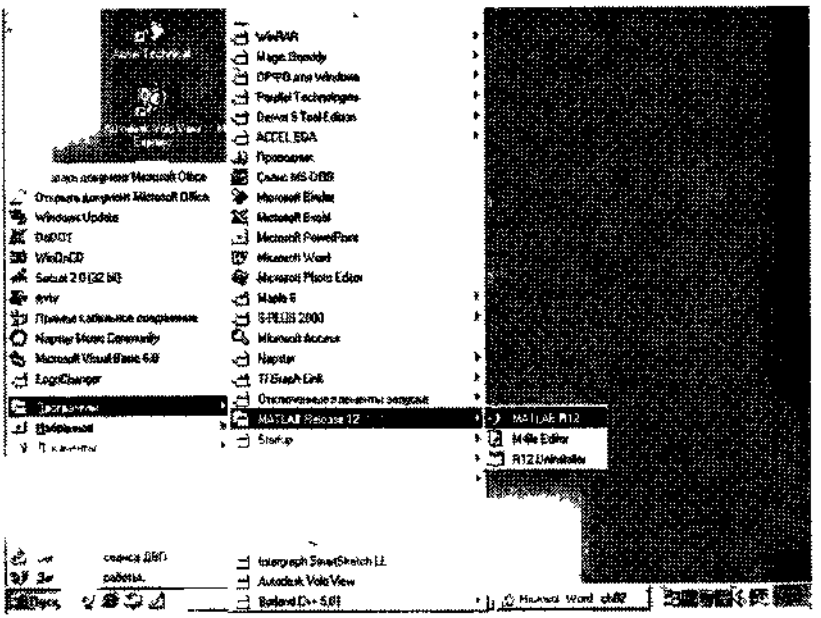


Рис. 1.2 Запуск MATLAB из главного меню Windows 98

После запуска появляется основное окно системы MATLAB, показанное на рис 1 3 Это обычное окно приложений Windows, его

можно перемещать, изменять в размерах, открывать на весь экран и т д

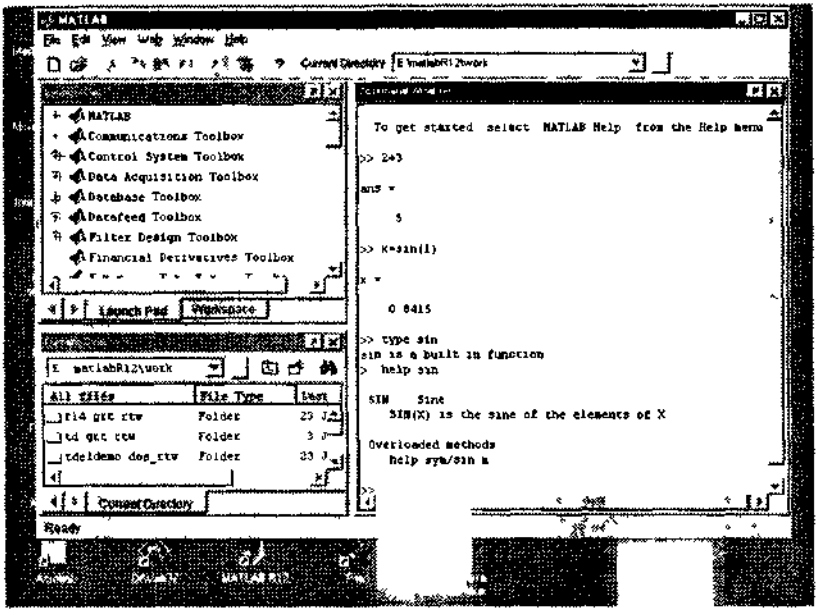


Рис. 1.3. Окно системы MATLAB после запуска и выполн ростых вычислений

Система готова к проведению вычислений в *командном режиме* Сеанс работы с MATLAB принято называть *сессией* (session) Можно сказать, что сессия является текущим документом, отражающим процесс работы пользователя с MATLAB Она содержит строки ввода, вывода и сообщения об ошибках

Панель инструментов и меню MATLAB 6.0

Интерфейс MATLAB 6 0 (рис 1 3) уже вполне отвечает современным канонам Он многооконный и имеет ряд средств прямого доступа к различным компонентам системы Появился новый пункт меню — Web, который дает прямой выход на web-страницу службы поддержки MATLAB и содержит другие полезные ссылки В панели инструментов (рис 1 4) добавлена кнопка восстановления ранее отмененной операции и меню просмотра файловой структуры с кнопкой его открытия

Рис. 1.4. Панель инструментов основного окна MATLAB

Кнопки панели инструментов выполняют следующие функции:

- New M-file — выводит пустое окно редактора m-файлов;
- Open file — открывает окно для загрузки m-файла;
- Cut — вырезает выделенный фрагмент и помещает его в буфер обмена;
- Copy — копирует выделенный фрагмент в буфер;
- Paste — вставляет фрагмент из буфера в текущую строку ввода;
- Undo — отменяет предшествующую операцию;
- Redo — восстанавливает последнюю отмененную операцию;
- Simulink — открывает окно браузера библиотек Simulink;
- Help — открывает окно справки.

Эти функции дублируются в очень простом меню системы MATLAB. Оно имеет следующие пункты:

- File — стандартные операции открытия нового файла, загрузки, сохранения и печати файлов;
- Edit — стандартные операции редактирования сессии и работы с буфером обмена (команды Cut, Copy и Paste);
- View — вывод и скрытие панели инструментов и управление видом интерфейса;
- Web — доступ к Интернет-ресурсам;
- Windows — установка свойств окна;
- Help — доступ к справочным подсистемам.

В левой части окна системы появились окна со вкладками Launch Pad/Workspace доступа к компонентам системы и вкладками текущей директории Current Directory и истории сессии History. Они обеспечивают оперативный контроль за состоянием системы.

Многие пользователи уже привыкли к крайней простоте интерфейса у прежних систем класса MATLAB. Учтя это, разработчики системы ввели в пункт меню View команду Desktop Layout ▸ Command Windows Only (Только командное окно). Стоит ее исполнить, как вид

окна системы станет очень напоминать добрый старый интерфейс версий MATLAB 5.* (рис. 1.5). Лишь пункт меню Web и меню файловой структуры в панели инструментов напоминают, что это окно новой версии MATLAB 6.0.

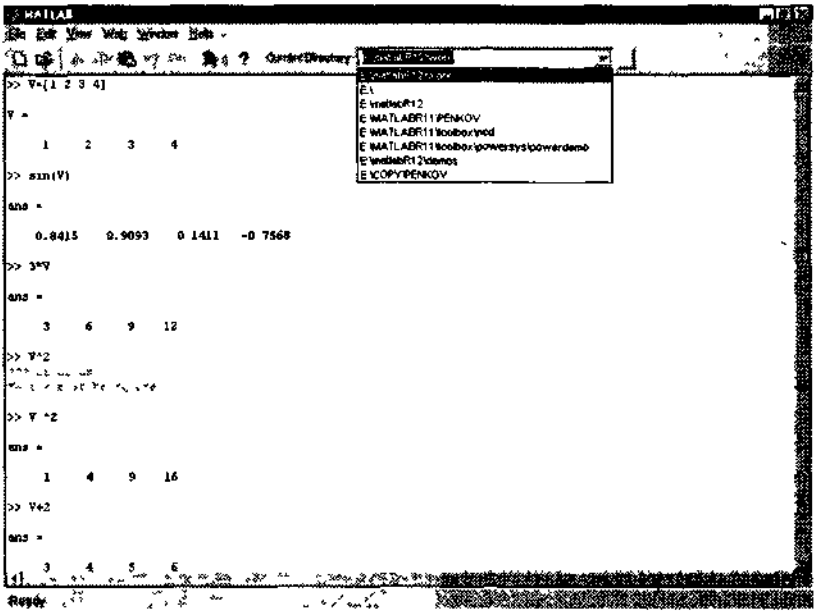


Рис. 1.5. Упрощенный интерфейс системы MATLAB 6.0

Если вы все же хотите использовать прелести нового интерфейса, то выполните в том же пункте меню команду Desktop Layout ▸ Default (Интерфейс по умолчанию). Там же вы найдете и другие возможности модификации вида интерфейса MATLAB 6.0.

Операции строчного редактирования

При работе с MATLAB в командном режиме действует простейший строчный редактор. Его команды перечислены в представленной ниже таблице.

Обратите особое внимание на применение клавиш ↑ и ↓. Они используются для подстановки после маркера строки ввода » ранее введенных строк из специального стека, например для их исправления, дублирования или дополнения. При этом указанные клавиши обес-

печивают перелистывание ранее введенных строк снизу вверх или сверху вниз.

Комбинация клавиш	Назначение
→ или Ctrl+b	Перемещение курсора вправо на один символ
← или Ctrl+f	Перемещение курсора влево на один символ
Ctrl+→ или Ctrl+r	Перемещение курсора вправо на одно слово
Ctrl+← или Ctrl+l	Перемещение курсора влево на одно слово
Home или Ctrl+a	Перемещение курсора в начало строки
End или Ctrl+e	Перемещение курсора в конец строки
↑ и ↓ или Ctrl+p и Ctrl+n	Перелистывание предыдущих команд вверх или вниз для подстановки в строку ввода
Del или Ctrl+d	Стирание символа справа от курсора
← или Ctrl+h	Стирание символа слева от курсора
Ctrl+k	Стирание до конца строки
Esc	Очистка строки ввода
Ins	Включение/выключение режима вставки
PgUp	Перелистывание страниц сессии вверх
PgDn	Перелистывание страниц сессии вниз

Команды управления окном

Перечислим некоторые команды управления окном командного режима:

- `clc` — очищает экран и размещает курсор в левом верхнем углу пустого экрана;
- `home` — возвращает курсор в левый верхний угол окна;
- `echo имя файла on` — включает режим вывода на экран текста Script-файла (файла-сценария);
- `echo имя файла off` — выключает режим вывода на экран текста Script-файла;
- `echo имя файла` — меняет режим вывода на противоположный;
- `echo on all` — включает режим вывода на экран текста всех m-файлов;
- `echo off all` — отключает режим вывода на экран текста всех m-файлов;

- more on — включает режим постраничного вывода (полезен при просмотре больших m-файлов);
- more off — отключает режим постраничного вывода (в этом случае для просмотра больших файлов надо пользоваться линейкой прокрутки).

В версии MATLAB 6.0 команды clc и home действуют одинаково — очищают экран и помещают курсор в левый верхний угол окна командного режима работы. Команды echo позволяют включать или выключать отображение текстов m-файлов. Для просмотра длинных листингов файлов полезно включить постраничный вывод командой more on.

MATLAB в режиме прямых вычислений

Система MATLAB создана таким образом, что любые (подчас весьма сложные) вычисления можно выполнять в режиме *прямых вычислений*, то есть без подготовки программы. Работа с системой в этом режиме носит диалоговый характер и происходит по правилу «задал вопрос — получил ответ» (см. рис. 1.3 и 1.5). Пользователь набирает на клавиатуре вычисляемое выражение, редактирует его (если нужно) в командной строке и завершает ввод нажатием клавиши ENTER.

Даже из приведенных простых примеров можно сделать некоторые полезные выводы:

- для указания ввода исходных данных используется символ »;
- данные вводятся с помощью простейшего строчного редактора;
- чтобы заблокировать вывод результата вычисления некоторого выражения, после него надо установить знак ; (точка с запятой);
- если не указана переменная для значения результата вычислений, то MATLAB назначает для этого переменную с именем ans;
- знаком присваивания является привычный математикам знак равенства =, а не комбинированный знак :=, как во многих других математических системах;
- встроенные функции (например, sin) записываются строчными буквами, а их аргументы указываются в *круглых скобках*;
- результат вычислений выводится в строках вывода (без знака »);
- диалог происходит в стиле «задал вопрос — получил ответ».

Следующий пример (см. рис. 1.5) иллюстрирует применение системы MATLAB для выполнения простых векторных операций. В этом примере задается четырехэлементный вектор *V* со значениями элементов 1, 2, 3 и 4. Далее (сосредоточьте на этом внимание!) вычисляются функции синуса и экспоненты с аргументом в виде *вектора*, а не скаляра.

Две записи для вектора — V=[1 2 3 4] и V=[1.2.3.4] — являются идентичными. Таким образом, векторы задаются списком своих элементов, разделяемых пробелами или запятыми. Список заключается в квадратные скобки. Для выделения *n*-го элемента вектора V используется выражение V(n). Оно задает соответствующую индексированную переменную.

В большинстве математических систем вычисление sin(V) и exp(V), где V — вектор, сопровождалось бы выдачей ошибки, поскольку функции sin и exp должны иметь аргумент в виде скалярной величины. Однако MATLAB — матричная система, а вектор является разновидностью матрицы размером 1 × n. Поэтому в нашем случае результат вычислений будет вектором того же размера, что и аргумент V, но элементы возвращаемого вектора будут синусами или экспонентами от элементов вектора V.

В дальнейшем мы будем приводить примеры в основном прямо в тексте книги. Например, сессия, представленная на рис. 1.3 и 1.5, будет записываться следующим образом:

```
To get started  select «MATLAB Help» from the Help menu
» 2+3
ans =
     5
» sin(1)
ans =
    0.8415
» type sin
sin is a built-in function
» help sin
  SIN       Sine
     SIN(X) is the sine of the elements of X
  Overloaded methods
     help sym/sin m
»
» V=[1 2 3 4]
V =
     1     2     3     4
» sin(V)
ans =
```

```
    0.8415        0.9093        0.1411       -0.7568
» 3*V
ans =
     3     6     9    12
» V^2
??? Error using ==> ^
Matrix must be square
» V .^2
ans =
     1     4     9    16
» V+2
ans =
     3     4     5     6
»
```

Из приведенных примеров видно, что матрица задается в виде ряда векторов, представляющих ее строки и заключенных в квадратные скобки. Для разделения элементов векторов используется пробел или запятая, а для отделения одного вектора от другого — точка с запятой. Для обозначения отдельного элемента матрицы `M` используется выражение вида `M(j,i)`, где `M` — имя матрицы, `j` — номер строки и `i` — номер столбца.

Строки ввода в текстах сессии будут отмечаться приглашением к вводу » в их начале. Отсутствие приглашения к вводу означает, что дается запись m-файла, то есть фрагмента программы. Ради компактности записи пустые строки будут опускаться.

О переносе строки в сессии

Длинные выражения можно переносить на другую строку с помощью знака многоточия « » (3 или более точек), например:

```
s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7
1/8 + 1/9 - 1/10 + 1/11 - 1/12.
```

Этот прием может оказаться весьма полезным для создания наглядных документов, у которых предотвращается попадание строк в невидимую область окна. Максимальное число символов в одной строке — 4096.

Основные объекты MATLAB

Понятие о математическом выражении

Центральным понятием всех математических систем является *математическое выражение*, например:

```
2+3
2 301*sin(x)
4+exp(3)/5
sqrt(y)/2
sin(pi/2)
```

Математические выражения строятся на основе чисел, констант, переменных, операторов, функций и специальных знаков. Ниже даются краткие пояснения сути этих понятий.

Действительные и комплексные числа

Число — простейший объект MATLAB, представляющий количественные данные. Числа можно считать константами, имена которых совпадают с их значениями. Числа используются в общепринятом представлении:

```
0
2
-3
2 301
0 00001
123 456e-24
-234 456e10
```

Как нетрудно заметить, в мантиссе чисел целая часть отделяется от дробной не запятой, а точкой, что принято в большинстве языков программирования. Для отделения порядка числа от мантиссы используется символ e. Знак «плюс» у чисел не ставится, а знак «минус» у числа называют *унарным минусом*. Пробелы между символами в числах не допускаются.

Числа могут быть *комплексными*: $z=\mathrm{Re}(x)+\mathrm{Im}(x)*i$. Такие числа содержат действительную $\mathrm{Re}(z)$ и мнимую $\mathrm{Im}(z)$ части. Мнимая часть имеет множитель *i* или *j*, означающей корень квадратный из –1:

```
3i
2j
2+3i
-3 141i
-123 456+2 7e-3i
```

Функция `real(z)` возвращает действительную часть комплексного числа, $\mathrm{Re}(z)$, а функция `imag(z)` — мнимую, $\mathrm{Im}(z)$. Для получения модуля комплексного числа используется функция `abs(z)`, а для вычисления фазы — `angle(Z)`. Ниже даны примеры работы с комплексными числами:

```
» i
ans =
          0 + 1.0000i
» j
ans =
          0 + 1.0000i
» z=2+3i
z =
     2.0000 + 3.0000i
» abs(z)
ans =
     3.6056
» real(z)
ans =
     2
» imag(z)
ans =
     3
» angle(z)
ans =
     0.9828
```

В MATLAB не принято делить числа на целые и дробные, короткие и длинные и т. д., хотя задавать числа в таких формах можно. Операции над числами выполняются с *двойной точностью*.

Форматы чисел

По умолчанию MATLAB выдает числовые результаты в *нормализованной форме* с четырьмя цифрами после десятичной точки и одной до нее. Для установки формата представления чисел используется команда

```
» format <имя формата>
```

Ниже представлены примеры изменения формата для числовых данных:

```
x=[4/3 1.2345e-6]
format short              1.3333                                  0.0000
format short e            1.3333E+000                         1.2345E-006
format long          1.333333333333338        0.000001234500000
format long e             1.333333333333338E+000
1.234500000000000E-006
format bank           1.33                               0.00
```

Задание формата сказывается только на *форме вывода* чисел. Вычисления все равно происходят с двойной точностью, а ввод чисел возможен в любом удобном для пользователя виде.

Константы и системные переменные

Константа — это предварительно определенное числовое или символьное значение, представленное уникальным именем. Числа (например, 1, –2 и 1.23) являются безымянными *числовыми константами.*

Другие виды констант в MATLAB принято назвать *системными переменными*, поскольку, с одной стороны, они задаются системой при ее загрузке, а с другой — могут переопределяться. Основные системные переменные, применяемые в системе MATLAB, указаны ниже:

- `i` или `j` — мнимая единица (корень квадратный из –1);
- `pi` — число $\pi = 3.1415926...$;
- `eps` — погрешность для операций над числами с плавающей точкой (2^{-52});
- `realmin` — наименьшее число с плавающей точкой (2^{-1022});
- `realmax` — наибольшее число с плавающей точкой (2^{1023});
- `inf` — значение машинной бесконечности;
- `ans` — переменная, хранящая результат последней операции и обычно вызывающая его отображение на экране;
- `NaN` — указание на нечисловой характер данных (Not-a-Number).

Вот примеры применения системных переменных:

```
» 2*pi
ans =
      6 2832
» eps
ans =
      2 2204e-016
» realmin
ans =
      2 2251e-308
» realmax
ans =
      1 7977e+308
» 1/0
Warning  Divide by zero
ans =
      Inf
» 0/0
Warning  Divide by zero
ans =
      NaN
```

Как отмечалось, системные переменные могут *переопределяться*. Можно задать системной переменной `eps` иное значение, например

eps=0.0001. Однако важно то, что их значения по умолчанию задаются сразу после загрузки системы. Поэтому неопределенными в отличие от обычных переменных системные переменные не могут быть никогда.

Символьная константа — это последовательность символов, заключенная в апострофы, например:

```
'Hello my friend!'
'Привет'
'2+3'
```

Если в апострофы помещено математическое выражение, то оно *не вычисляется* и рассматривается просто как цепочка символов. Так что ввод '2+3' не будет возвращать число 5. Однако с помощью специальных функций преобразования символьные выражения могут быть переведены в вычисляемые.

Текстовые комментарии

Поскольку MATLAB используется для достаточно сложных вычислений, важное значение имеет наглядность их описания. Она достигается, в частности, с помощью текстовых комментариев. *Текстовые комментарии* вводятся с помощью символа %, например так:

```
%It is factorial function
```

ПРИМЕЧАНИЕ

В MATLAB 6 есть проблемы с символами кириллицы, например наблюдается перевод строки при вводе буквы «с» русского алфавита. К счастью, при подготовке m-файлов в редакторе-отладчике (он будет описан позже) этой проблемы не возникает.

Обычно первые строки m-файлов служат для описания их назначения и выводятся на экран после команды

```
» help <имя_файла>
```

Считается правилом хорошего тона вводить в m-файлы достаточно подробные текстовые комментарии. Без таких комментариев даже разработчик программных модулей быстро забывает о сути собственных решений.

Переменные и присваивание им значений

Переменные — это имеющие имена объекты, способные хранить некоторые данные. В зависимости от этих данных переменные

могут быть числовыми или символьными, векторными или матричными.

В системе MATLAB можно задавать переменным определенные значения. Для этого используется операция *присваивания*, вводимая знаком равенства =:

```
Имя_переменной = Выражение
```

Типы переменных заранее не декларируются. Они определяются выражением, значение которого присваивается переменной. Так, если это выражение — вектор или матрица, то переменная будет векторной или матричной.

Имя переменной (ее *идентификатор*) может содержать сколько угодно символов (кроме специальных), но запоминается и идентифицируется только 41 начальный символ. Имя любой переменной не должно совпадать с именами других переменных, функций и процедур системы, то есть оно должно быть уникальным. Имя может содержать буквы, цифры и символ подчеркивания, но должно начинаться с буквы.

Желательно использовать содержательные имена для обозначений переменных, например speed_1 для переменной, обозначающей скорость первого объекта. Переменные могут быть обычными и *индексированными*, то есть элементами векторов или матриц (см. выше). Могут использоваться и *символьные* переменные, причем символьные значения заключаются в апострофы, например s='Demo'.

Дефрагментация рабочего пространства

По мере задания одних переменных и стирания других рабочее пространство перестает быть непрерывным и начинает содержать «дыры» и всякий «мусор». Во избежание потерь памяти при работе с объемными данными (а векторы, матрицы и массивы относятся к таковым) следует использовать команду pack, осуществляющую дефрагментацию рабочего пространства.

Уничтожение определений переменных

Для очистки рабочего пространства в командном режиме используется функция clear в разных формах, например:

- clear — уничтожение определений всех переменных;
- clear x — уничтожение определения переменной x;

- `clear a, b, c` — уничтожение определений нескольких переменных.

Уничтоженная (стертая в рабочем пространстве) переменная становится неопределенной. Использовать такие переменные нельзя — подобные попытки будут сопровождаться выдачей сообщений об ошибке.

Операторы и функции

Оператор — это специальное обозначение для определенной операции над данными — *операндами*. Например, простейшими арифметическими операторами являются знаки суммы +, вычитания -, умножения * и деления /. Операторы используются совместно с операндами. Например, в выражении 2+3 знак + является оператором сложения, а числа 2 и 3 — операндами.

Следует отметить, что большинство операторов относится к матричным операциям, что может служить причиной серьезных недоразумений. Например, операторы умножения * и деления / вычисляют произведение и частное двух массивов, векторов или матриц. Есть ряд специальных операторов, например оператор \ означает деление *справа налево*, а операторы * и ./ означают *почленное* умножение и деление массивов.

Следующие примеры поясняют сказанное на примере операций с векторами:

```
» V1=[2 4 6 8]
V1 =
     2     4     6     8
» V2=[1 2 3 4]
V2 =
     1     2     3     4
» V1/V2
ans =
     2
» V1 *V2
ans =
     2     8    18    32
» V1 /V2
ans =
     2     2     2     2
```

Полный список операторов можно получить, используя команду

```
» help ops
```

Приведем только часть полного списка операторов, содержащую арифметические операторы:

```
» help ops
 Operators and special characters

 Arithmetic operators
      Plus          - Plus                                  +
      uplus         - Unary plus                            +
      minus         - Minus                                 -
      uminus        - Unary minus                           -
      mtimes        - Matrix multiply                       *
      times         - Array multiply                        *
      mpower        - Matrix power                          ^
      power         - Array power                           .^
      mldivide      - Backslash or left matrix divide       \
      mrdivide      - Slash or right matrix divide          /
      ldivide       - Left array divide                     .\
      rdivide       - Right array divide                    ./
      kron          - Kronecker tensor product              kron
```

Функции — это имеющие уникальные имена объекты, выполняющие определенные преобразования над своими аргументами и при этом возвращающие результаты этих преобразований. *Возврат результата* — отличительная черта функций. При этом результат вычисления функции с одним выходным параметром подставляется на место ее вызова, что позволяет использовать функции в математических выражениях, например `2*sin(pi/2)`.

Функции в общем случае имеют список аргументов (параметров), заключенный в круглые скобки. Например, функция Бесселя записывается как `bessel(NU,X)`. В данном случае список параметров содержит два аргумента — `NU` в виде числа и `X` в виде вектора. Многие функции допускают формы записи, отличающиеся списком параметров. Если функция возвращает несколько значений, то она записывается в виде

```
[Y1, Y2,   ]=func(X1  X2    )
```

где `Y1, Y2,` — список *выходных* аргументов и `X1, X2,` — список *входных* аргументов (параметров).

Со списком элементарных функций можно ознакомиться, выполнив команду `help elfun`, а со списком специальных функций — с помощью команды `help specfun`. Функции могут быть *встроенными* (внутренними) и *внешними*, или *m-функциями*. Так, встроенными являются наиболее распространенные элементарные функции, например `sin(x)` и `exp(y)`, тогда как функция `sinh(x)` является внешней функцией. Определения внешних функций содержатся в m-файлах. Задание таких функций с помощью специального редактора m-файлов мы рассмотрим в разделе «Работа с файлами»

данной главы. Встроенные функции хранятся в откомпилированном ядре системы MATLAB, в силу чего они выполняются предельно быстро. Всего MATLAB содержит около 600 различных функций.

Применение оператора : (двоеточие)

Очень часто бывает необходимо формирование упорядоченных числовых последовательностей. Такие последовательности нужны для создания векторов или значений абсциссы при построении графиков. Для этого в MATLAB используется оператор (двоеточие):

```
Начальное_значение Шаг Конечное_значение
```

Данная конструкция порождает возрастающую последовательность чисел, которая начинается с начального значения, идет с заданным шагом и завершается конечным значением. Если шаг не задан, то он принимает значение 1. Если конечное значение указано меньшим, чем начальное значение, — выдается сообщение об ошибке. Примеры применения оператора даны ниже:

```
» 1 5
ans =
     1     2     3     4     5
» i=0 2 10
i =
      0     2     4     6     8    10
» j=10 -2 2
j =
     10     8     6     4     2
» V=0 pi/2 2*pi.
» V
V =
      0    1 5708    3 1416    4 7124    6 2832
» X=1 - 2 0
X =
    1 0000    0 8000    0 6000    0 4000    0 2000
         0
» 5 2

ans =

     Empty matrix  1-by-0
```

Как отмечалось, принадлежность MATLAB к матричным системам вносит коррективы в назначение операторов и приводит при их неумелом использовании к казусам. Рассмотрим следующий характерный пример:

```
» x=0:5
x =
     0     1     2     3     4     5
» cos(x)
ans =
    1.0000    0.5403   -0.4161   -0.9900   -0.6536
    0.2837
» sin(x)/x
ans =
   -0.0862
```

Вычисление массива косинусов здесь прошло корректно. А вот вычисление массива значений функции $\sin(x)/x$ дает на первый взгляд неожиданный эффект — вместо массива с шестью элементами вычислено единственное значение.

Причина «парадокса» здесь в том, что оператор / вычисляет отношение двух *массивов*. Если они одной размерности, то результат будет одним числом, что в данном случае и выдала система. Чтобы действительно получить массив значений $\sin(x)/x$, надо использовать специальный оператор *почленного* деления массивов — ./. Тогда будет получен массив чисел:

```
» sin(x)./x
Warning: Divide by zero.
ans =
       NaN    0.8415    0.4546    0.0470   -0.1892
  -0.1918
```

Впрочем, и тут без особенностей не обошлось. Так, при $x = 0$ значение $\sin(x)/x$ дает устранимую неопределенность вида 0/0=1. Однако, как и всякая численная система, MATLAB классифицирует попытку деления на 0 как ошибку и выводит соответствующее предупреждение. А вместо ожидаемого численного значения выводится символьная константа NaN, означающая, что неопределенность 0/0 — это все же не обычное число.

Выражения с оператором : могут использоваться в качестве аргументов функций для получения их множественных значений, например:

```
x=1/2.
» bessel(0:1:5.1/2)
ans =
          0.9385          0.2423          0.0306          0.0026
0.00020.0000
» bessel(0.0.1:5)
ans =
    1.0000    0 7652    0 2239   -0.2601   -0 3971
  -0.1776
```

Таким образом, оператор : является весьма удобным средством задания регулярной последовательности чисел. Он широко используется при работе со средствами построения графиков. В дальнейшем, в разделе «Работа со средствами графики», мы расширим представление о возможностях этого оператора.

Диагностика ошибок

Важное значение в диалоге с системой MATLAB имеет *диагностика ошибок*. MATLAB проверяет вводимые команды и выражения и выдает соответствующие сообщения об ошибках или предупреждения. Пример вывода сообщения об ошибке (деление на 0) только что приводился.

Рассмотрим еще ряд примеров. Введем, к примеру, ошибочное выражение

```
» sqr(2)
```

и нажмем клавишу ENTER. Система сообщит об ошибке:

```
??? Undefined function or variable 'sqr'.
```

Это сообщение говорит о том, что не определена переменная или функция, и указывает, какая именно — sqr. В данном случае, разумеется, можно просто набрать правильное выражение. Однако в случае громоздкого выражения лучше воспользоваться редактором. Для этого достаточно нажать клавишу ↑ для перелистывания предыдущих строк. В результате в строке ввода появится выражение

```
» sqr(2)
```

с курсором в конце. Теперь с помощью клавиши ← следует установить курсор после буквы r и нажать клавишу T, а затем клавишу ENTER. Выражение примет следующий вид:

```
» sqrt(2)
ans =
          1.4142
```

Теперь вычисления дают ожидаемый результат — значение квадратного корня из двух.

Приведем еще один пример:

```
» hsin(1)
??? Undefined function or variable 'hsin'.
» sinh(1)
ans =
      1.1752
```

В этом примере система сообщает, что функция или переменная с именем hsin не определена ни как внутренняя, ни как m-функция. Зато далее мы видим, что в составе функций системы MATLAB найдена функция с именем sinh — она задана в виде m-файла.

Иногда в ходе вывода результатов вычислений появляется сокращение NaN, обозначающее неопределенность, например, вида 0/0 или Inf/Inf, где Inf — системная переменная со значением машинной бесконечности. Могут появляться и различные предупреждения об ошибках (на английском языке). Например, при делении на 0 конечного числа появляется предупреждение Warning: Divide by Zero. (Внимание: Деление на ноль). Диапазон чисел, представимых в системе, — от 10^{-308} до 10^{+308}.

Вообще говоря, в MATLAB следует отличать *предупреждение* об ошибке от *сообщения* о ней. *Предупреждения* (обычно после слова Warning) не останавливают вычисления и лишь предупреждают пользователя о том, что диагностируемая ошибка способна повлиять на ход вычислений. *Сообщение* об ошибке (после знаков ???) останавливает вычисления.

Операции с векторами и матрицами

Особенности задания векторов и матриц

Система MATLAB по умолчанию предполагает, что каждая заданная переменная — это вектор или матрица. Все определяется конкретным значением переменной. Например, если задано X=1, то это значит, что X есть вектор с единственным элементом, имеющим значение 1. Если надо задать вектор из трех элементов, то их значения следует перечислить в квадратных скобках, разделяя пробелами или запятыми:

```
» V=[1 2 3]
V =
     1 2 3
```

Задание матрицы требует указания нескольких строк. Для различения строк используется знак ; (точка с запятой):

```
» M=[1 2 3; 4 5 6; 7 8 9];
» M
M =
     1     2     3
     4     5     6
     7     8     9
```

Возможен ввод элементов матриц и векторов в виде арифметических выражений, содержащих любые доступные системе функции, например:

```
» V= [2+2/(3+4) exp(5) sqrt(10)];
» V
V =
    2.2857    148.4132    3.1623
```

Для указания отдельного элемента вектора или матрицы используются выражения вида `V(i)` или `M(i, j)`, например:

```
» M(2, 2)
ans =
     5
```

Если нужно присвоить элементу `M(i, j)` новое значение x, следует использовать выражение

```
M(i,j)=x
```

Например, если элементу `M(2, 2)` надо присвоить значение 10, следует записать:

```
» M(2, 2)=10 .
```

Выражение `M(i)` с одним индексом дает доступ к элементам матрицы, развернутым в один столбец. Такая матрица образуется из исходной, если подряд выписать ее столбцы. Следующий пример поясняет такой доступ к элементам матрицы `M`:

```
» M=[1 2 3; 4 5 6; 7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
» M(2)
ans =
     4
» M(8)
ans =
     6
» M(9)
ans =
     9
» M(5)=100;
» M
M =
     1     2     3
     4   100     6
     7     8     9
```

Возможно задание векторов и матриц с комплексными элементами, например:

```
» 1=sqrt(-1);
» CM = [1 2; 3 4] + 1*[5 6; 7 8]
```

или

```
» CM = [1+5*1 2+6*i; 3+7*i 4+8*i]
```

Это создает матрицу:

```
CM =
      1.0000 + 5.0000i        2.0000 + 6.0000i
      3.0000 + 7.0000i        4.0000 + 8.0000i
```

Для поэлементных операций над массивами перед знаком операции ставится точка. Например, оператор * означает умножение для векторов или матриц, а оператор .* — почленное умножение всех элементов массива. Так, если M — матрица, то M.*2 даст матрицу, все элементы которой умножены на скаляр — число 2. Впрочем, для умножения матрицы на скаляр оба выражения — M*2 и M.*2 — оказываются равноценными.

Имеется также ряд особых функций для задания векторов и матриц. Например, функция magic(n) задает магическую матрицу размером $n \times n$, у которой сумма всех столбцов, всех строк и даже диагоналей равна одному и тому же числу:

```
» M=magic(4)
M =
      16      2      3      13
      5      11      10      8
      9      7      6      12
      4      14      15      1
» sum(M)
ans =
      34      34      34      34
» sum(M')
ans =
      34      34      34      34
» sum(diag(M))
ans =
      34
» M(1,2)+M(2,2)+M(3,2)+M(4,2)
ans =
      34
```

Запись M' означает транспонирование матрицы, то есть замену строк столбцами.

Объединение малых матриц в большую

Описанный способ задания матриц позволяет выполнить операцию *конкатенации* — объединения малых матриц в большую. Например, создадим вначале магическую матрицу размером 3 × 3:

```
» A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
```

Теперь можно построить матрицу, содержащую четыре матрицы:

```
» B=[A A+16;A+32 A+16]
B =
     8     1     6    24    17    22
     3     5     7    19    21    23
     4     9     2    20    25    18
    40    33    38    24    17    22
    35    37    39    19    21    23
    36    41    34    20    25    18
```

Полученная матрица имеет уже размер 6×6.

Удаление столбцов и строк матриц

Для формирования матриц и выполнения ряда матричных операций возникает необходимость удаления отдельных столбцов и строк матрицы. Для этого используются пустые квадратные скобки [ ]. Проделаем это с матрицей M:

```
» M=[1 2 3; 4 5 6; 7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
```

Удалим второй столбец, используя оператор : (двоеточие):

```
» M(:,2)=[ ]
M =
     1     3
     4     6
     7     9
```

А теперь, используя оператор : (двоеточие), удалим вторую строку:

```
» M(2,:)=[ ]
M =
     1     3
     7     9
```

Операции сессии

Браузер рабочего пространства

В памяти компьютера переменные занимают определенное место, называемое *рабочим пространством* (workspace). В левой части основного окна MATLAB 6.0 можно найти вкладку браузера рабочего пространства. На рис. 1.6 эта вкладка показана в деталях, после ряда операций с переменными, векторами и матрицами.

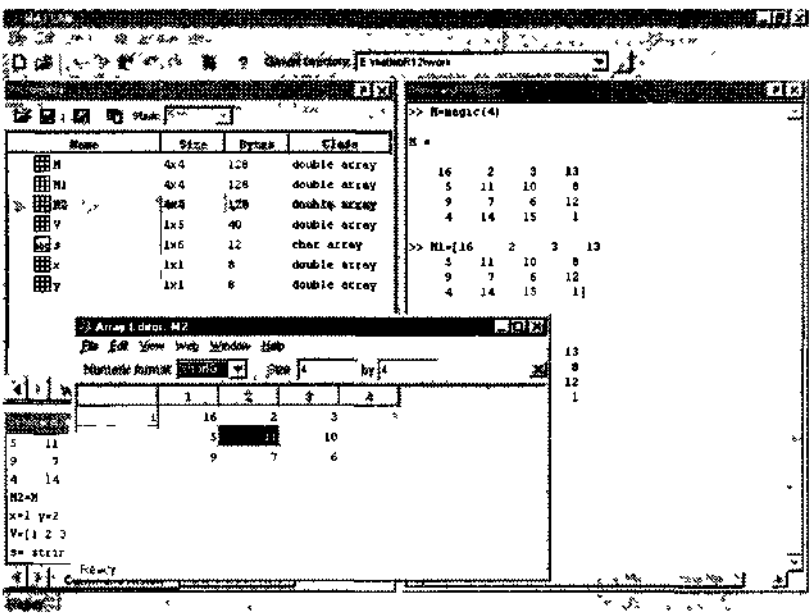


Рис. 1.6. Работа с браузером рабочего пространства

Как нетрудно заметить, окно браузера дает наглядную информацию о типе переменных и их размере (в байтах). Активизация той или иной переменной выводит окно ее редактирования — на рис. 1.6 показано такое окно для матрицы M2.

Браузер истории сессии

MATLAB 6.0 имеет очень полезный новый инструмент — браузер истории сессии (Command History). Это список всех команд, кото-

рые использовались в сессиях системы до последней очистки окна браузера. Рисунок 1.7 показывает работу с окном браузера истории сессии, которое видно в левом нижнем углу окна системы MATLAB.

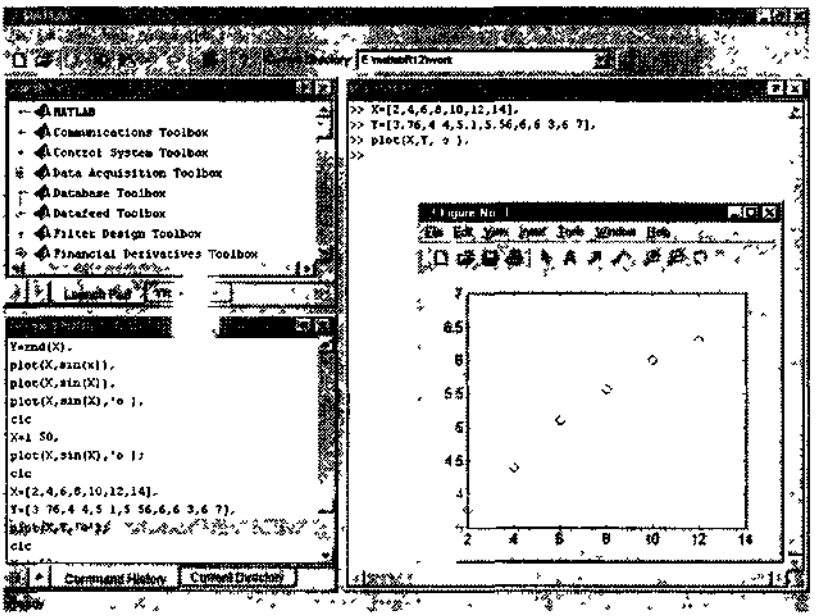


Рис. 1.7. Работа с браузером истории сессии

Вместо весьма специфической и не всегда удобной прокрутки команд в сессии браузер истории команд дает вполне современное средство просмотра. Указав курсором мыши выбранную команду и дважды щелкнув левой кнопкой мыши, можно перенести команду в командную строку основного окна MATLAB. Именно так на рис. 1.7 в окно сессии перенесены команды задания векторов X и Y и затем команда построения заданных в них точек.

Сохранение рабочего пространства сессии

Переменные и определения новых функций в системе MATLAB хранятся в особой области памяти, именуемой рабочим пространством. MATLAB позволяет сохранять значения переменных в виде бинарных файлов с расширением .mat. Для этого служит команда

save Возможно использование слова save и в формате функции, а не команды, например:

```
save('fname','var1' 'var2')
```

В этом случае имена файлов и переменных задаются строковыми константами.

Ведение дневника

Сессии не записываются на диск стандартной командой save. Однако, если такая необходимость есть, можно воспользоваться специальной командой для ведения так называемого *дневника* сессии:

- diary file_name — включает режим записи на диск всех команд в строках ввода и полученных результатов в виде текстового файла с указанным именем;
- diary off — приостанавливает запись в файл;
- diary on — возобновляет запись в файл.

Следующий пример поясняет технику применения команды diary:

```
» diary myfile m
» 1+2
ans =
      3
» diary off
» 2+3
ans =
      5
» diary on
» sin(1)
ans =
      0 8415
» diary off
```

Нетрудно заметить, что в данном примере первая операция — 1 + + 2 = 3 — будет записана в файл myfile.m, вторая — 2 + 3 = 5 — не будет записана, третья операция — sin(1) = 0.8415 — снова будет записана. Таким образом, будет создан файл-сценарий (script-файл), который можно просмотреть командой type:

```
» type myfile
1+2
ans =
      3
diary off
sin(1)
ans =
```

```
     0 8415
diary off
```

Рекомендуется записывать такие файлы с расширением, отличным от .m, например .txt. Это позволит встраивать файлы дневника сессии в документы, содержащие ее описание.

Загрузка рабочего пространства сессии

Для загрузки рабочего пространства ранее проведенной сессии (если она была сохранена) можно использовать команду `load имя файла` или `load('fname')`.

Если команда (или функция) `load` используется в ходе сессии, то произойдет замена текущих значений переменных на те значения, которые были сохранены в считываемом mat-файле.

Для задания имен загружаемых файлов может использоваться символ *, означающий загрузку всех файлов с определенными свойствами. Например, `load demo* mat` означает загрузку всех файлов, имена которых начинаются с demo, например demo1, demo2, demoa, demob и т. д. Имена загружаемых файлов можно формировать с помощью операций над строковыми выражениями.

Завершение вычислений

Иногда из-за ошибок в программе или из-за сложности решаемой задачи MATLAB «зацикливается» и перестает выдавать результаты либо непрерывно выдает их, хотя в этом уже нет необходимости. Для прерывания вычислений в этом случае достаточно нажать одновременно клавиши Ctrl и C (латинская).

Завершение работы с системой

Прерывание вычислений, например в случае зацикливания, осуществляется нажатием совместно клавиш Ctrl и C (латинская). Для завершения работы с системой можно использовать команды `quit`, `exit` или комбинацию клавиш Ctrl+Z. Если необходимо сохранить значения всех переменных (векторов, матриц) системы, то перед этим следует выполнить команду `save` с нужными параметрами. Команда `load` после загрузки системы считывает значения этих переменных и позволяет начать работу с системой с того момента, когда она была прервана.

Работа с файлами

Браузер компонентов системы MATLAB

Система MATLAB состоит из многих тысяч файлов, находящихся в множестве папок. Полезно иметь представление о содержании основных папок, поскольку это позволяет быстро оценить возможности системы, например узнать, какие операторы, функции или графические команды входят в систему.

В MATLAB особое значение имеют файлы двух типов — с расширениями .mat и .m. Первые являются бинарными файлами, в которых могут храниться значения переменных. Вторые представляют собой текстовые файлы, содержащие внешние программы, определения команд и функций системы. Именно к ним относится большая часть команд и функций, в том числе задаваемых пользователем для решения своих специфических задач. Нередко встречаются и файлы с расширением .c (на языке C), файлы с откомпилированными кодами с расширением .mex и другие. Исполняемые файлы имеют расширение .exe.

Особое значение имеет папка MATLAB/TOOLBOX, которая содержит основные компоненты системы MATLAB+Simulink. Окно браузера компонентов системы Launch Pad видно на рис. 1.7 сверху и слева. Работа с ним не требует никаких пояснений.

Стандарные m-файлы системы

Не менее важное значение имеет папка MATLAB. В ней содержится набор стандартных m-файлов системы. Ниже перечислены основные подпапки с этими файлами (деление на категории условно, на самом деле все подпапки находятся в общей папке MATLAB/TOOLBOX/MATLAB).

- Команды общего назначения:
  - General – команды общего назначения: работа со справкой, управление окном MATLAB, взаимодействие с операционной системой и т. д.
- Операторы, конструкции языка и системные функции:
  - ops — операторы и специальные символы;
  - lang — конструкции языка программирования;
  - strfun — строковые функции;

- iofun — функции ввода/вывода;
- timefun — функции времени и дат;
- datatypes — типы и структуры данных.

○ Основные математические и матричные функции:

- elmat — команды создания элементарных матриц и операций с ними;
- elfun — элементарные математические функции;
- specfun — специальные математические функции;
- matfun — матричные функции линейной алгебры;
- datafun — анализ данных и преобразование Фурье;
- polyfun — полиномиальные функции и функции интерполяции;
- funfun — функции функций и функции решения обыкновенных дифференциальных уравнений;
- soarfun — функции разреженных матриц.

○ Команды графики:

- graph2d — команды двумерной графики;
- graph3d — команды трехмерной графики;
- specgraph — команды специальной графики;
- graphics — команды дескрипторной графики;
- uitools — графика пользовательского интерфейса.

Полный состав файлов каждой подпапки (их список содержится в файле contents.m) можно просмотреть с помощью команды help <имя папки>.

Браузер файловой структуры

Работа с файлами существенно облегчается благодаря еще одной новинке MATLAB 6 — браузеру файловой структуры. Его вкладка находится в левом нижнем углу основного окна MATLAB. На рис. 1.8 показана работа с этим браузером, причем его окно теперь занимает всю левую часть экрана.

Выделив в списке какой-либо m-файл и дважды щелкнув левой кнопкой мыши, можно загрузить его в редактор-отладчик (окно редактора-отладчика показано на рис. 1.8 в центре экрана).

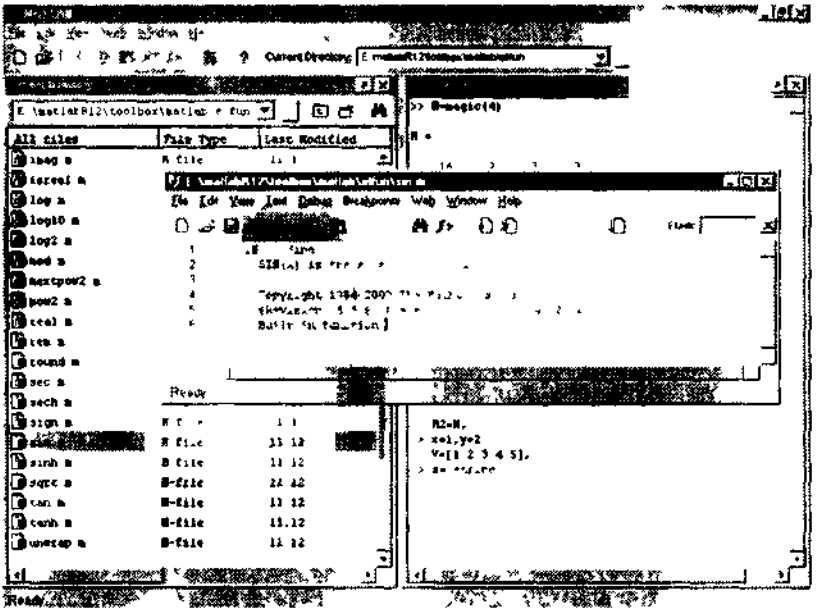


Рис. 1.8. Работа с браузером файловой структуры

Редактор-отладчик m-файлов

Многооконный редактор-отладчик с пустым окном редактирования m-файлов можно вызвать командой edit из командной строки или командой меню Edit ▸ New ▸ M-file. После этого в окне редактора можно создать свой файл, а также пользоваться средствами его отладки и запуска. Для запуска файла его необходимо записать на диск, используя команду Save as в меню File редактора.

На рис. 1.9 показано окно редактора-отладчика с текстом простого файла в окне редактирования и отладки. После подготовки файла используется команда Save As.

После записи файла на диск можно заметить, что команда Run в меню Tools редактора становится активной (до записи файла на диск она пассивна) и позволяет произвести запуск файла. Запустив команду Run, можно наблюдать исполнение m-файла — в нашем случае это построение рисунка в графическом окне (рис. 1.10) и вывод надписи о делении на ноль в ходе вычисления функции $\sin(x)/x$ в командном окне системы.

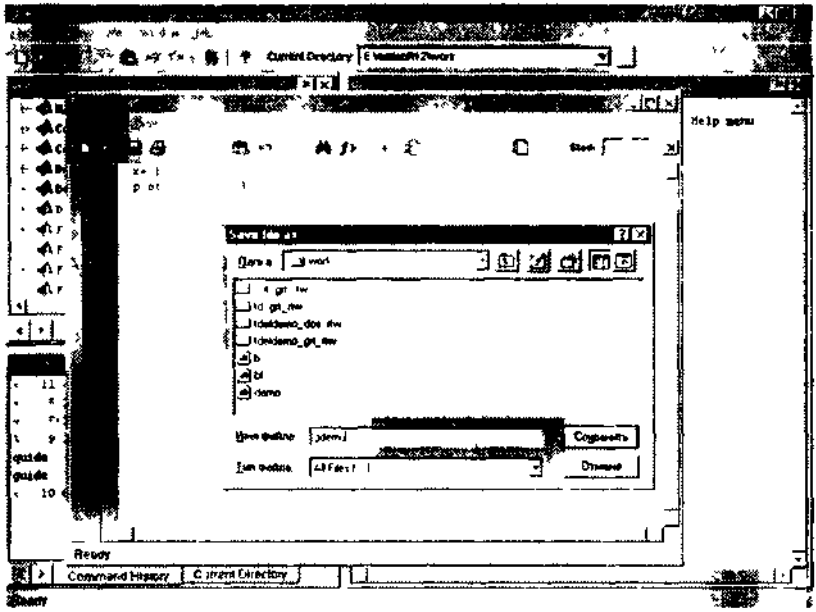


Рис. 1.9. Редактор отладчик файлов и запись файла на диск

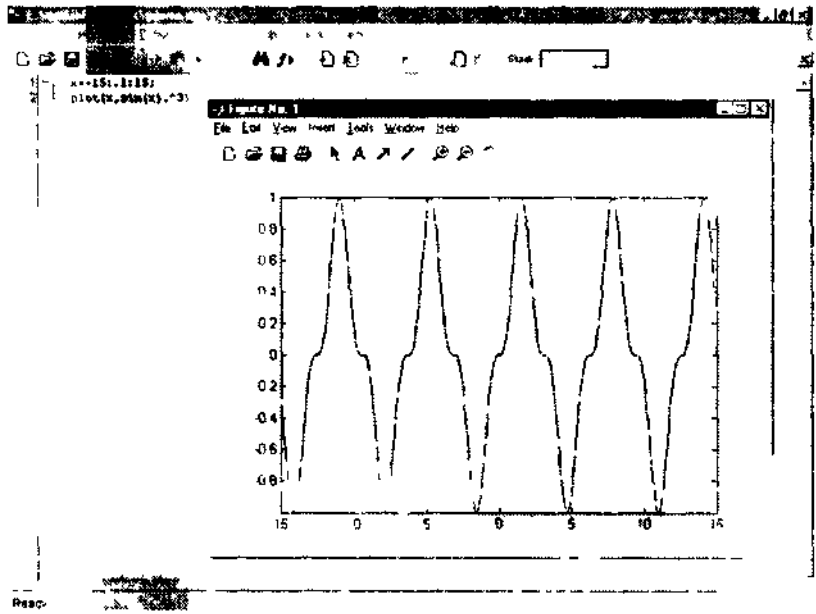


Рис. 1.10. Исполнение m-файла, показанного на рис 1 9

Для удобства работы с редактором-отладчиком строки программы в нем последовательно нумеруются. Редактор является многооконным; окно каждой программы оформляется как вкладка.

Цветовые выделения и проверка синтаксиса

Редактор-отладчик m-файлов выполняет синтаксическую проверку программного кода по мере ввода текста. При этом используются следующие цветовые выделения:

- ключевые слова языка программирования — синий цвет;
- операторы, константы и переменные — черный цвет;
- комментарии после знака % — зеленый цвет;
- символьные переменные (в апострофах) — зеленый цвет;
- синтаксические ошибки — красный цвет.

Благодаря цветовым выделениям вероятность синтаксических ошибок резко снижается.

Однако далеко не все ошибки диагностируются. Ошибки, связанные с неверным применением операторов или функций (например, применение оператора - вместо + или функции `cos(x)` вместо `sin(x)` и т. д.), не способна обнаружить ни одна система программирования. Устранение такого рода ошибок (их называют семантическими) — дело пользователя, отлаживающего свои программы.

Файлы-сценарии

М-файлы, создаваемые редактором-отладчиком, делятся на два класса:

- файлы-сценарии (Script-файлы), не имеющие входных параметров;
- файлы-функции, имеющие входные параметры.

Файл, показанный на рис. 1.9 и 1.10, является файлом-сценарием. Данный файл не имеет списка входных параметров и является примером простой процедуры без параметров. Он использует *глобальные переменные*, то есть такие переменные, значения которых могут быть изменены в любой момент сеанса работы и в любом месте программы. Структура этого файла следующая:

```
%Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями
```

Основным комментарием является первая строка текстовых комментариев, а дополнительным — последующие строки. Основной комментарий выводится при выполнении команд `lookfor` и `help <имя каталога>`. Полный комментарий выводится при выполнении команды `help <имя файла>`.

Для запуска файла-сценария из командной строки MATLAB достаточно указать его имя (рис. 1.11). Обратите внимание на команду `grid on`, выполняемую после запуска созданного файла. Эта команда наносит на график сетку из точечных линий.

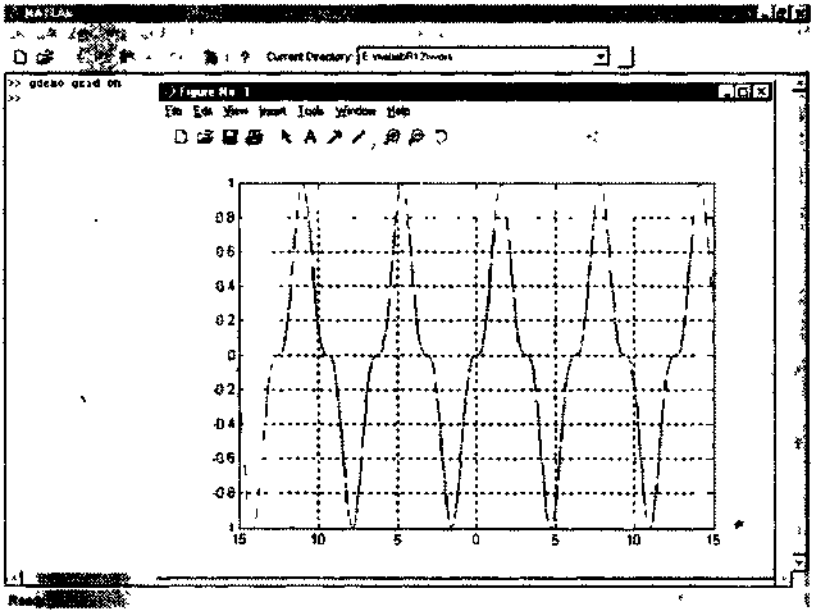


Рис. 1.11. Выполнение файла-сценария из командной строки

Тело файла может содержать любые математические и логические выражения, а также управляющие структуры (например, циклы и условные выражения), которые присущи языку программирования системы MATLAB. Он реализует модульный и объектно-ориентированный подход к подготовке программ и имеет средства для создания элементов интерфейса пользователя.

Файлы-функции

Файл-*функция* является типичным объектом — модулем — языка программирования системы MATLAB. Структура такого модуля с одним выходным параметром выглядит следующим образом:

```
function var=f_name(Список_параметров)
%Основной комментарий
%Дополнительный комментарий
Тело файла с любыми выражениями
var=выражение
```

Файл-функция имеет следующие свойства:

- он начинается с объявления function, после которого указывается имя переменной — выходного параметра, имя самой функции и список ее входных параметров;
- функция возвращает свое значение и может использоваться в виде f_name(Список_параметров) в математических выражениях;
- все переменные, имеющиеся в теле файла-функции, являются *локальными*, то есть действуют только в пределах тела функции;
- файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;
- правила вывода комментариев те же, что у файлов-сценариев;
- файл-функция служит средством расширения системы MATLAB;
- при обнаружении файла-функции он компилируется и затем исполняется, а созданные машинные коды хранятся в рабочем пространстве системы MATLAB.

Последняя конструкция var=выражение вводится, если требуется, чтобы функция возвращала результат вычислений. Если выходных параметров больше, то они указываются в квадратных скобках после слова function. При этом структура модуля имеет следующий вид:

```
function [var1,var2,...]=f_name(Список_параметров)
%Основной комментарий
%Дополнительный комментарий
Тело функции с любыми выражениями
var1=выражение
var2=выражение
.
```

Как уже отмечалось, переменные в файлах-функциях — локальные. Использование глобальных переменных в программных модулях имеет свои побочные эффекты. Применение локальных переменных

устраняет эти неудобства и отвечает требованиям структурного программирования.

Однако передача данных из модуля в модуль в этом случае происходит только через входные и выходные параметры, что требует тщательного планирования такой передачи. Команда `global var1 var2...` позволяет объявить переменные модуля функции глобальными. Таким образом, внутри функции могут использоваться и глобальные переменные, если это нужно по условиям решения вашей задачи.

В MATLAB можно включать *подфункции*. Они объявляются и записываются в теле основных функций и имеют идентичную им конструкцию:

```
function [mean,stdev] = statv(x)
%STATV Interesting statistics.
%Пример функции с встроенной подфункций
n = length(x);
mean = avg(x,n);
stdev = sqrt(sum((x-avg(x,n)).^2)/n);

%-----------
function m = avg(x,n)
%Mean subfunction
m = sum(x)/n;
```

В этом примере среднее значение элементов вектора `x` вычисляется с помощью подфункции `avg(x,n)`, тело которой записано в теле основной функции `statv`. Пример использования функции `statv` представлен ниже:

```
» V=[1 2 3 4 5]
V =
     1     2     3     4     5
» [a,m]=statv(V)
a =
     3
m =
     1.4142
» statv(V)
ans =
     3
» help statv
  STATV Interesting statistics.
  Пример функции с встроенной подфункций
```

Подфункции определены и действуют локально, то есть только в пределах m-файла, определяющего основную функцию. Так что заданные в некотором m-файле подфункции нельзя использовать ни в командном режиме работы, ни в других m-файлах. Команда `help`

<имя функции> выводит комментарий, относящийся только к основной функции, тогда как команда type <имя файла> выводит весь листинг m-файла.

Панель инструментов редактора и отладчика

Редактор-отладчик m-файлов имеет свое меню и свою инструментальную панель (рис. 1.12).

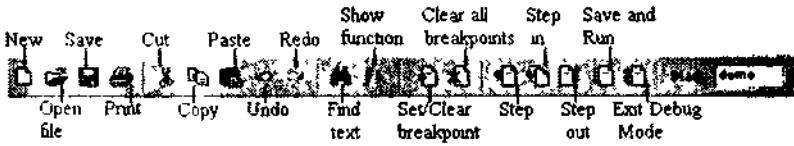


Рис. 1.12. Панель инструментов редактора-отладчика m-файлов

Назначение ее кнопок следующее:

- New — создание нового m-файла;
- Open — загрузка файла;
- Save — запись файла на диск;
- Print — печать содержимого текущего окна редактора;
- Cut — перенос выделенного фрагмента в буфер обмена;
- Copy — копирование выделенного объекта в буфер;
- Paste — размещение фрагмента из буфера в позиции текстового курсора;
- Undo — отмена предшествующей операции;
- Redo — повтор отмененной операции;
- Find text — нахождение указанного текста;
- Show function — показ функции;
- Set/Clear Breakpoint — установка/сброс точки прерывания;
- Clear All Breakpoints — сброс всех точек прерывания;
- Step — выполнение шага трассировки;
- Step In — пошаговая трассировка с заходом в вызываемые m-файлы;
- Step Out — пошаговая трассировка без захода в вызываемые m-файлы;

- Save and Run — сохранение и запуск;
- Exit Debug Mode — завершение отладки.

Работа с точками прерывания

Основным приемом отладки m-файлов является установка в их тексте точек прерывания (breakpoints). Они устанавливаются (и сбрасываются) с помощью кнопки Set/Clear Breakpoint. Сброс всех точек прерывания обеспечивается кнопкой Clear All Breakpoints.

Рассмотрим рис. 1.13, на котором показана конструкция цикла в окне редактора-отладчика. Как будет меняться переменная s, значение которой должно давать ряд натуральных чисел? Прежде всего для отладки надо записать программу на диск, а затем установить напротив выражения s=s+1 точку прерывания — она обозначена кружком. Для установки точки прерывания необходимо поместить текстовый курсор напротив указанного выражения и нажать кнопку Set/Clear Breakpoint.

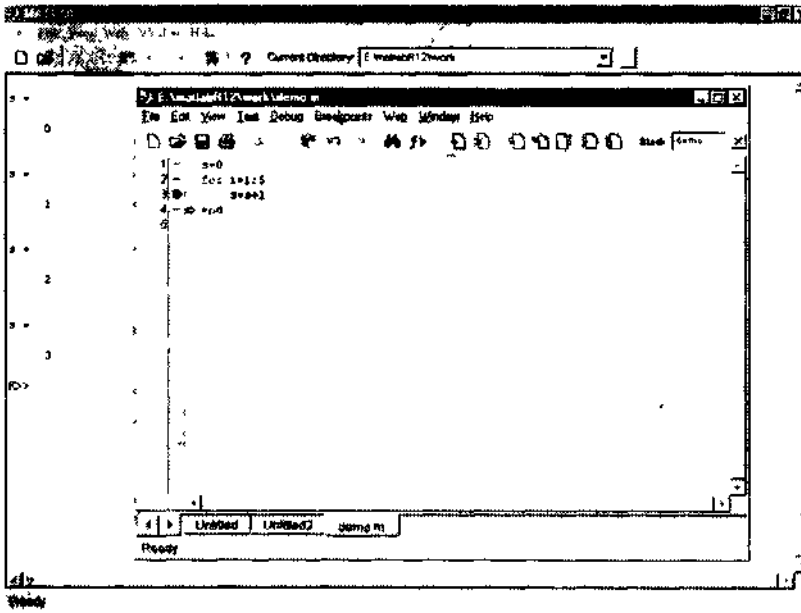


Рис. 1.13. Простейший пример применения точки прерывания в программе

Теперь при запуске программа будет выполнена до точки прерывания, после чего текущие значения s будут выведены в окне MATLAB.

С помощью кнопки Step можно выполнить очередной шаг вычислений и т. д. Если отпала необходимость останова в точках прерывания, достаточно кнопкой Clear All Breakpoints удалить сразу все точки прерывания. Желтая стрелка указывает, в каком месте программы произошел останов. Обратите внимание на то, что в этом примере каждый шаг исполнения цикла фиксируется в окне командного режима MATLAB.

При остановке в точке прерывания вы можете проконтролировать значения переменных как «вручную» (введя имя переменной в командной строке и нажав клавишу Enter), так и с помощью организации просмотра перед точкой прерывания. Вы можете задать выполнение программы без захода (кнопка Step Out) и с заходом в вызываемые m-файлы (кнопка Step In). Кнопка Exit Debug Mode прекращает операции отладки.

Работа со средствами графики

Обзор интерфейса графических окон

Средства графики системы MATLAB позволяют создавать множество графических окон, подобных показанному на рис. 1.14. Однако размещение графики в окне сессии не предусмотрено. Это возможно в специальном расширении Notebook, позволяющем встраивать объекты MATLAB (тексты, строки ввода и вывода, графики) в документы популярного текстового редактора Word 95/97.

В меню Edit окна графики наряду со стандартными операциями работы с буфером обмена есть ряд новых команд:

- Copy Figure — копирование фигуры (графика) в буфер;
- Copy Options — копирование опций графика;
- Figure Properties — вывод окна свойств графика;
- Axes Properties — вывод окна свойств осей графика;
- Current Object Properties — вывод окна свойств текущего объекта.

Для вывода свойств графиков, их осей и текущих объектов используется окно свойств графиков с соответствующими вкладками. На рис. 1.14 оно показано на переднем плане и иллюстрирует операции форматирования графика, показанного на заднем плане того же рисунка. Показан момент форматирования стиля линии двумерного графика и задания ей синего цвета. Возможно также форматирование осей и иных объектов графиков.

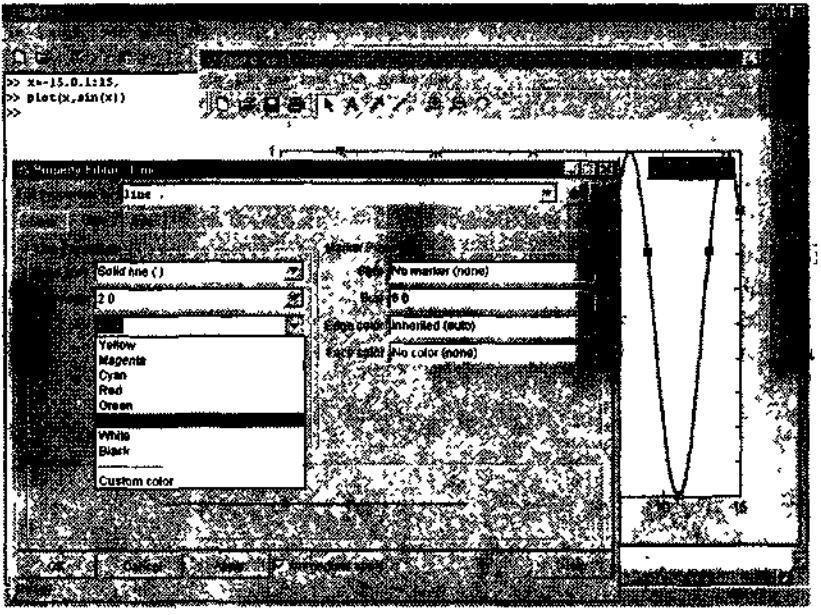


Рис. 1.14. Графическое окно MATLAB и окно свойств графики

С помощью панели инструментов, назначение кнопок которой вполне очевидно, можно открывать пустое окно графики, загружать в окно график и выводить его на печать, менять режим редактирования графика, наносить на графики надписи, стрелки и линии, увеличивать и уменьшать масштаб графика и вращать его мышью.

Панель инструментов камеры обзора

Отличительной особенностью окна графики в версии MATLAB 6.0 стало появление второй панели инструментов со средствами форматирования трехмерной графики. Эта панель (рис. 1.15) выводится командой View ▸ Camera Toolbar.

Форматирование трехмерных объектов происходит при помощи воображаемой фотокамеры, через которую наблюдается объект. Действие кнопок панели инструментов продублировано командами меню Tools:

- Edit Plot — редактирование свойств графика;
- Zoom In — увеличение масштаба графика;

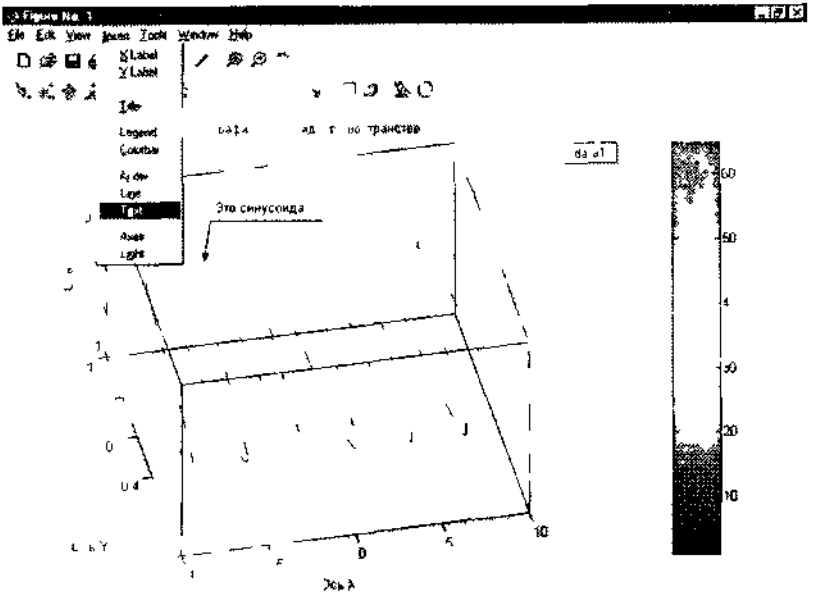


Рис 1 15 Пример форматирования и вращения графика

- Zoom Out — уменьшение масштаба графика,
- Rotate 3D — вращение в пространстве,
- Move Camera — установка камеры обзора
- Camera Motion — установка перемещения камеры обзора,
- Camera Axes — установка координатных осей при работе с камерой
- Camera Reset — сброс установок камеры
- Basic Fiting — проведение аппроксимации и регрессии,
- Data Statistics — получение статистических данных для точек графика

Операции вставки

В пункте Insert меню графического окна предусмотрены операции вставки основных объектов надписей по осям титульной надписи надписи внутри рисунка стрелки, отрезка прямой легенды и шкалы цветов На рис 1 15 даны примеры их применения

Специальные средства графики

Обработка данных в графическом окне

В MATLAB 6 в пункте Tools графического окна появились две новые команды

- Basic Fiting — основные виды аппроксимации (регрессии),
- Data Statistics — статистические параметры данных

Команда Basic Fiting открывает окно, дающее доступ к ряду видов аппроксимации и регрессии сплайновой, эрмитовой и полиномиальной со степенями от 1 (линейная аппроксимация) до 10, в том числе со степенью 2 (квадратичная аппроксимация) и 3 (кубическая аппроксимация). Команда Data Statistics открывает окно с результатами простейшей статистической обработки данных.

Полиномиальная регрессия для табличных данных

Пусть некая зависимость $y(x)$ задана векторами координат ее точек

```
>> X [2 4 6 8 10 12 14]
>> Y=[3 7.6 4 4 5 1 5 5.6 6 6 3 6 7]
>> plot(X Y  o )
```

Рисунок 1.16 показывает пример выполнения полиномиальной регрессии (аппроксимации) для степеней полинома 1, 2 и 3. Иными словами, выполняется линейная, параболическая и кубическая регрессия.

ВНИМАНИЕ

При проведении полиномиальной аппроксимации надо помнить, что максимальная степень полинома на 1 меньше числа точек, то есть числа элементов в векторах X и Y.

Поясним, что же показано на рис. 1.16. В окне сессии MATLAB видна запись исходных векторов и команды построения заданных ими точек кружками. Выполнив команду Tools Basic Fitting можно получить окно регрессии (окно Basic Fitting). В этом окне отмечены флажками три упомянутых выше вида полиномиальной регрессии. Установка флажка Show equations выводит в графическом окне записи уравнений регрессии.

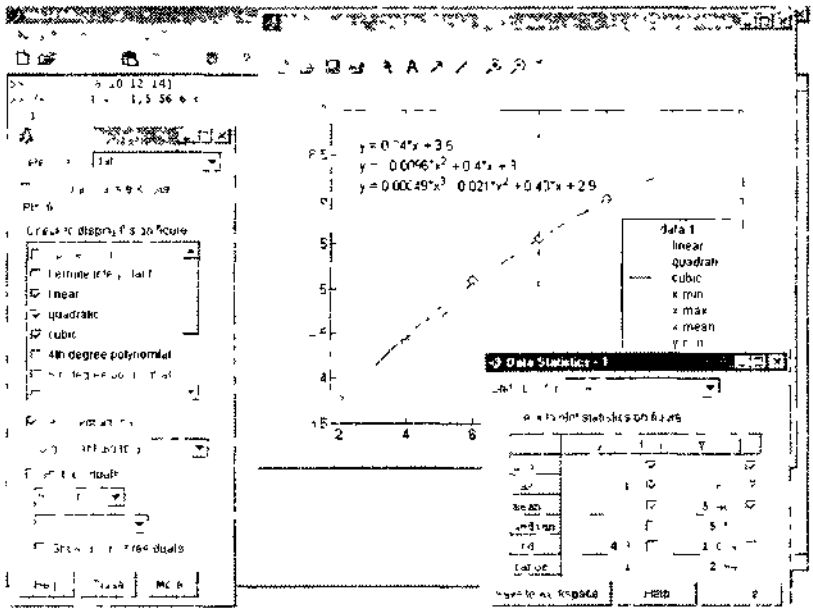


Рис. 1.16. Пример обработки табличных данных в графическом окне

По команде Tools Data Statistics выводится окно Data Statistics с рядом статистических параметров для данных, представленных векторами X и Y. Отметив флажком тот или иной параметр в этом окне (рис. 1.16), можно наблюдать соответствующие построения на графике, например построение вертикалей с минимальным, средним и максимальным значением y и горизонталей с минимальным, средним и максимальным значением x.

Оценка погрешности аппроксимации

Средства обработки данных графического окна позволяют строить столбчатый или линейчатый графики погрешностей в узловых точках и наносить на эти графики норму погрешности. Норма дает статистическую оценку среднеквадратической погрешности. Для вывода графика погрешности надо установить флажок Plot residuals и в меню под этим флажком выбрать тип графика (рис. 1.17).

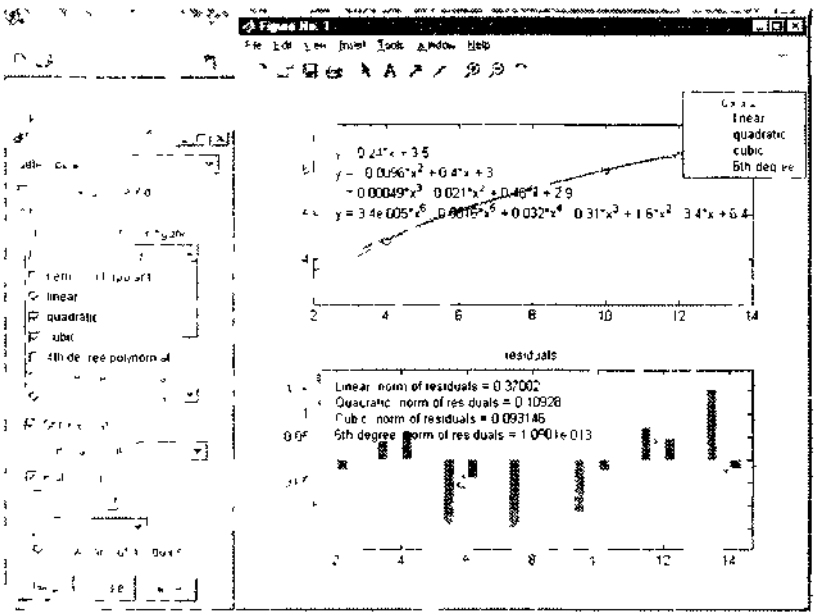


Рис. 1.17. Пример вывода данных обработки со столбчатым графиком погрешности

На рис. 1.17 приведены данные по полиномиальной аппроксимации степени 1, 2, 3 и 6. Последний случай предельный, поскольку максимальная степень полинома должна быть на 1 меньше числа точек (их 7). В этом случае регрессия вырождается в обычную (без статистической обработки) полиномиальную аппроксимацию. Тогда линии графика аппроксимирующей функции проходят через узловые точки, а погрешность в этих точках равна 0 (точнее, ничтожно мала).

Рисунок 1.18 демонстрирует построение графика погрешности отрезками линий. Кроме того, опцией Separate figure задано построение графика погрешности в отдельном окне — оно расположено под графиком узловых точек и функций аппроксимации.

Таким образом, интерфейс графического окна позволяет выполнять эффективную обработку данных наиболее распространенными способами.

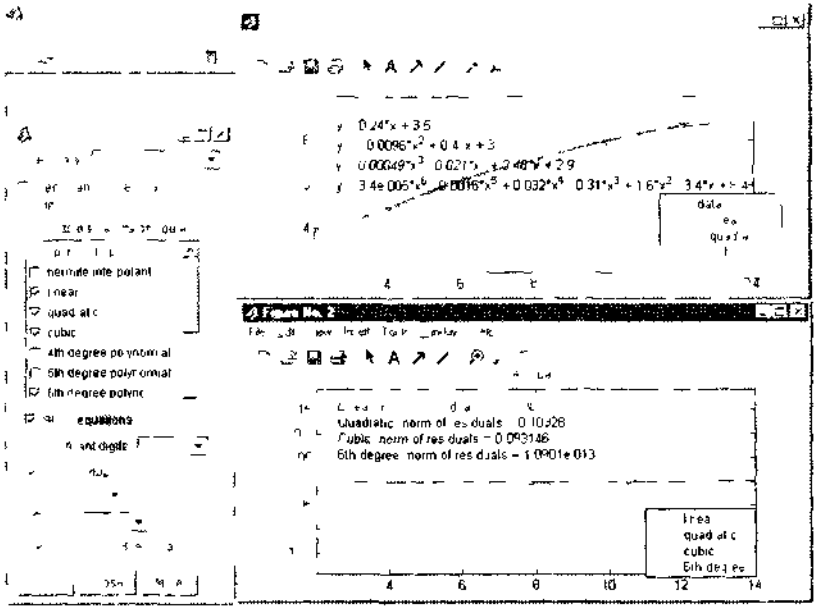


Рис. 1.18 Пример обработки табличных данных с выводом графиков погрешности в отдельном окне

Сплайновая и эрмитова интерполяции в графическом окне

Теперь рассмотрим пример сплайновой интерполяции в графическом окне. Зададим 50 точек синусоидальной функции, как это показано в левом верхнем углу рис. 1.19.

Как видно из рис. 1.19, при построении исходной функции по точкам невозможно судить о ее форме. Точки оказываются разбросанными по полю рисунка, и создается впечатление (кстати, абсолютно ложное) случайности их расположения. Попытка аппроксимации полиномом 8-й степени не дает положительного результата — кривая проходит внутри облака точек, совершенно не связывая их.

Однако картина кардинально меняется, стоит применить сплайновую интерполяцию, которая является одним из видов многоинтервальной интерполяции. На этот раз кусочная линия интерполяции (рис. 1.20) прекрасно проходит через все точки и поразительно напоминает синусоиду. Даже ее пики со значениями 1 и –1 воспроизводятся удивительно точно, причем даже в случаях, когда на них нет узловых точек.

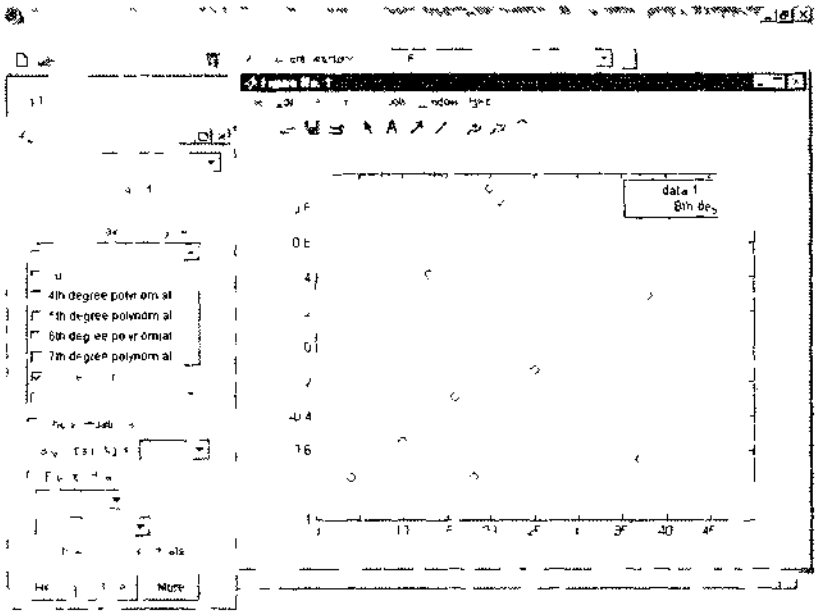


Рис. 1.19 Пример аппроксимации синусоиды в графическом окне

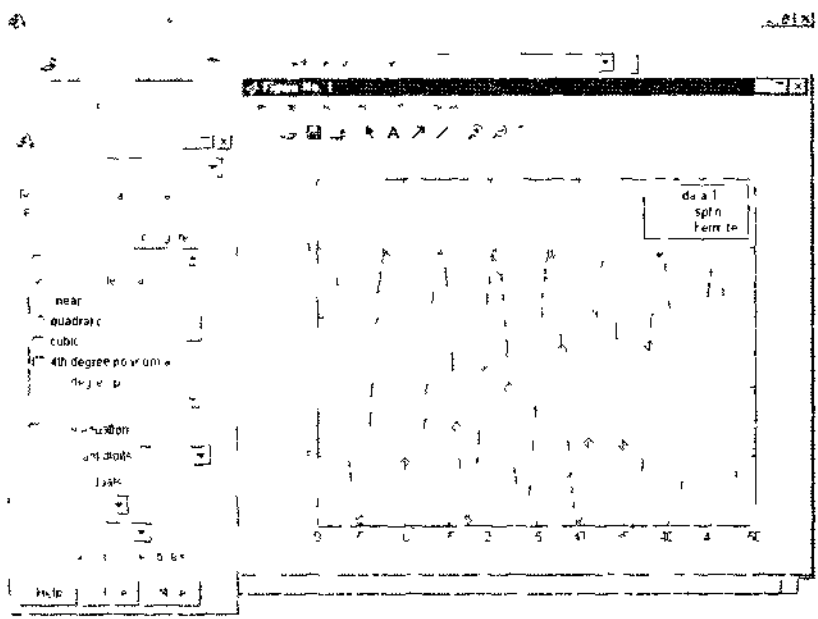


Рис. 1.20 Примеры сплайновой и эрмитовой интерполяции в графическом окне

Причина столь великолепного результата кроется в известных особенностях сплайновой интерполяции: она выполняется по трем ближайшим точкам, причем эти тройки точек постепенно перемещаются от начала точечного графика функции к ее концу. Кроме того, непрерывность первой и второй производных при сплайновой интерполяции делает кривую очень плавной, что характерно и для первичной функции — синусоиды. Так что данный пример просто является удачным случаем применения сплайновой интерполяции.

MATLAB 6.0 дает возможность использовать в графическом окне еще один вид многоинтервальной интерполяции на основе полиномов третьей степени Эрмита. Техника интерполяции здесь та же, что и в случае сплайновой интерполяции. Это показывает рис. 1.20, где дан пример и эрмитовой интерполяции. Полиномы Эрмита имеют более гибкие линии, чем сплайны. Они точнее следуют за отдельными изгибами исходной зависимости, что хорошо видно из рис. 1.20.

Мы не можем практически называть этот подход полноценной аппроксимацией, поскольку в данном случае нет единого выражения для аппроксимирующей функции. На каждом отрезке (интервале) приближения используется кубический полином с новыми коэффициентами. Поэтому и вывода аппроксимирующей функции в поле графика не предусмотрено.

Можно сделать вывод, что сплайновая интерполяция лучше, когда нужно эффективное сглаживание быстро меняющихся от точки к точке данных и когда исходная зависимость описывается линиями, которые мы наблюдаем при построении их с помощью гибкой линейки. Эрмитова интерполяция лучше отслеживает быстрые изменения исходных данных, но имеет худшие сглаживающие свойства.

Графики разного типа в одном окне

Бывает, что в одном окне надо расположить несколько координатных осей с различными графиками без наложения их друг на друга. Для этого используются команды:

- `subplot` — создает новые объекты класса `axes` (подокна);
- `subplot(m,n,p)` или `subplot(mnp)` — разбивает графическое окно на $m \times n$ подокон, при этом `m` — число подокон по горизонтали, `n` — число подокон по вертикали, а `p` — номер подокна, в которое бу-

дет выводиться текущий график (подокна отсчитываются последовательно по строкам);

- subplot(H), где H — дескриптор для объекта axes, предоставляет альтернативный способ задания подокна для текущего графика;
- subplot('position',[left bottom width height]) — создает подокно с заданными нормализованными координатами (в пределах от 0 0 до 1 0);
- subplot(111) и clf reset — удаляют все подокна и возвращают графическое окно в обычное состояние.

Следующий пример иллюстрирует применение команды subplot:

```
» x=-5 0 1 5.
subplot(2 2 1),plot(x,sin(x))
subplot(2 2 2),plot(sin(5*x),cos(2*x+0 2))
subplot(2 2 3) contour(peaks)
subplot(2 2 4) surf(peaks)
```

В этом примере последовательно строится четыре графика различного типа, размещаемых в разных подокнах (рис. 1.21).

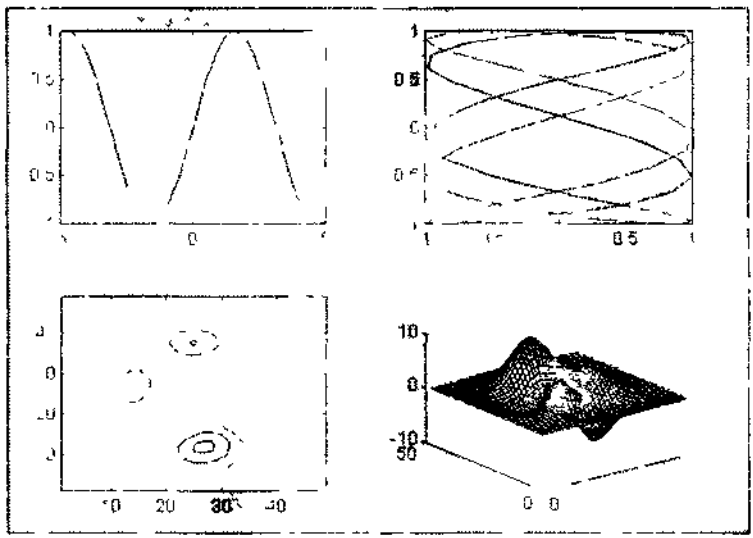


Рис. 1.21. Четыре графика различного типа в одном окне

Следует отметить, что для всех графиков возможна индивидуальная установка дополнительных объектов, например титульных надписей, надписей по осям и т. д.

Низкоуровневая дескрипторная графика

Дескрипторная (описательная) графика является низкоуровневой Она создается с помощью графических объектов, имеющих свойства наследования и иерархию, показанную на рис 1 22 Названия объектов говорят о их сути

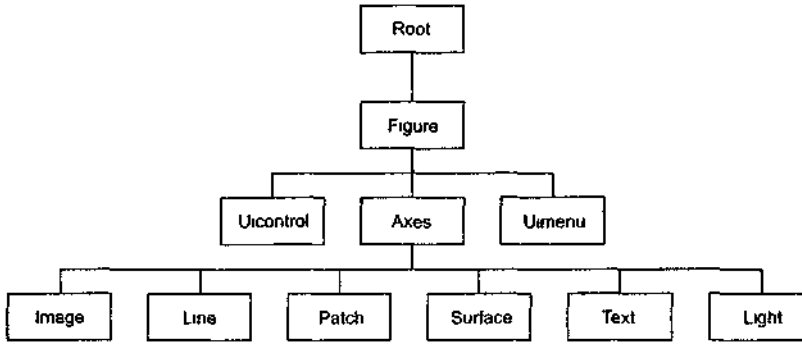


Рис. 1.22. Иерархия объектов дескрипторной графики системы MATLAB

Для иллюстрации возможностей дескрипторной графики создадим файл ms1 m

```
[x y] = meshgrid([ 2  4 2])
Z =sin(x ^2+y ^2)
fh = figure( Position  [350 275 400 300]  Color   w )
ah = axes( Color  [ 8  8  8]  XTick  [-2  1 0 1 2]
 YTick  [ 2  1 0 1 2])
sh = surface( XData  x  YData  y  ZData  Z
 FaceColor  get(ah  Color )+ 1
 EdgeColor   k   Marker   o
 MarkerFaceColor  [ 5 1  85])
```

В этом файле заданы три объекта прямоугольник fh — объект класса figure, оси с метками ah — объект класса axes и трехмерная поверхность sh — объект класса surface Теперь создадим второй фаил — ms2.m

```
h(1) = axes( Position  [0 0 1 1]) sphere
h(2) = axes( Position  [0 0  4  6]) peaks
h(3) = axes( Position  [0  5  5  5]) sphere
h(4) = axes( Position  [ 5 0  4  4]) sphere
h(5) = axes( Position  [ 5  5  5  3]) cylinder([0 0 0 5])
set(h  Visible   off )
set(gcf  Renderer   painters )
```

Здесь задано 5 трехмерных объектов три сферы разных размеров, поверхность peaks и цилиндр Запустив последовательно эти фай-

лы, получим сложный многокомпонентный объект, представленный на рис 1 23

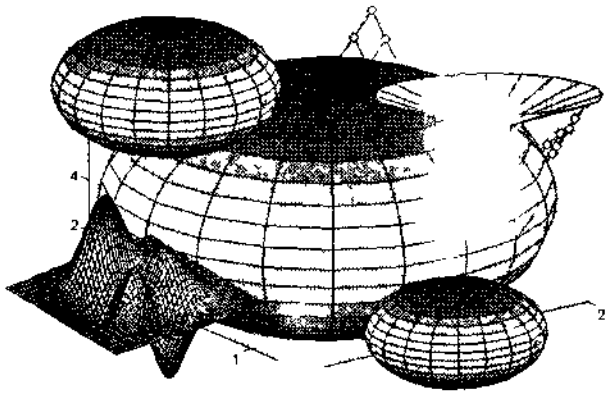


Рис. 1.23. Комбинированный рисунок полученный при запуске файлов ms1 и ms2

Следует отметить, что дескрипторная графика рассчитана не столько на конкретных пользователей, работающих с MATLAB как с прикладной программой, сколько на опытных разработчиков программного обеспечения для этой системы

Глава 2. Первое знакомство с Simulink 4.0

- Основные возможности пакета Simulink 4.0
- Запуск Simulink и основы работы с пакетом
- Запуск моделей Simulink из среды MATLAB
- Моделирование аттрактора Лоренца
- Решение дифференциальных уравнений Ван-дер-Поля
- Работа с редактором дифференциальных уравнений
- Моделирование кубика с пружинкой
- Моделирование системы терморегулирования дома
- Моделирование работы унитаза
- Моделирование механических систем с анимацией
- Применение логических блоков средств контроля данных
- Общие замечания по моделированию систем

Основные возможности пакета Simulink 4.0

Назначение пакета

В состав системы MATLAB 6.0 входит пакет моделирования динамических систем — Simulink 4.0. Это новая существенно доработанная версия популярного пакета, который уже давно считается одним из лучших пакетов моделирования блочно заданных динамических систем.

Пакет Simulink является ядром интерактивного программного комплекса, предназначенного для *математического моделирования* ли-

нейных и нелинейных динамических систем и устройств, представленных своей функциональной блок-схемой, именуемой *S-моделью*, или просто *моделью*. При этом возможны различные варианты моделирования: во временной области, в частотной области, с событийным управлением, на основе спектральных преобразований Фурье, с использованием метода Монте-Карло (реакция на воздействия случайного характера) и т. д.

Для построения функциональной блок-схемы моделируемых устройств Simulink имеет обширную *библиотеку* блочных компонентов и удобный *редактор блок-схем*. Он основан на графическом интерфейсе пользователя и по существу является типичным средством *визуально-ориентированного программирования*. Используя *палитры компонентов* (наборы), пользователь с помощью мыши переносит нужные блоки с палитр на рабочий стол пакета Simulink и соединяет линиями входы и выходы блоков. Таким образом создается блок-схема системы или устройства, то есть модель.

Simulink автоматизирует следующий, наиболее трудоемкий этап моделирования: он составляет и решает сложные системы алгебраических и дифференциальных уравнений, описывающих заданную функциональную схему (модель), обеспечивая удобный и наглядный визуальный контроль за поведением созданного пользователем *виртуального устройства*. Вам достаточно уточнить (если нужно) вид анализа и запустить Simulink в режиме *симуляции* (откуда и название пакета — Simulink) созданной модели системы или устройства. В дальнейшем мы будем использовать термин «моделирование».

Средства визуализации результатов моделирования в пакете Simulink настолько наглядны, что порой создается ощущение, что созданная в виде блок-схемы модель работает «как живая». Более того, Simulink практически мгновенно меняет математическое описание модели по мере ввода ее новых блоков, даже в том случае, когда этот процесс сопровождается сменой порядка системы уравнений и ведет к существенному качественному изменению поведения системы. Впрочем, это является одной из главных целей пакета Simulink.

Ценность Simulink заключается и в обширной, открытой для изучения и модификации библиотеке компонентов (блоков). Она включает источники сигналов с практически любыми временными зависимостями, масштабирующие, линейные и нелинейные

преобразователи с разнообразными формами передаточных характеристик, квантующее устройство, интегрирующие и дифференцирующие блоки и т. д.

В библиотеке имеется целый набор виртуальных регистрирующих устройств — от простых измерителей типа вольтметра или амперметра до универсальных осциллографов, позволяющих просматривать временные зависимости выходных параметров моделируемых систем — например, токов и напряжений, перемещений, давлений и т. п. Имеется даже графопостроитель для создания фигур в полярной системе координат, например фигур Лиссажу и фазовых портретов колебаний. Simulink имеет средства анимации и звукового сопровождения. А в дополнительных библиотеках можно отыскать и такие «дорогие приборы», как анализаторы спектра сложных сигналов, многоканальные самописцы и средства анимации графиков.

Программные средства моделирования динамических систем известны давно, к ним относятся, например, программы Tutsim и LabVIEW for Industrial Automation. Однако для эффективного применения таких средств необходимы высокоскоростные решающие устройства. Интеграция одной из самых быстрых матричных математических систем — MATLAB 6.0 — с пакетом Simulink 4.0 открывает новые возможности использования самых современных математических методов для решения задач динамического и ситуационного моделирования сложных систем и устройств.

Средства графической анимации Simulink позволяют строить виртуальные физические лаборатории с наглядным представлением результатов моделирования. Возможности Simulink охватывают задачи математического моделирования сложных динамических систем в физике, электро- и радиотехнике, в биологии и химии — словом, во всех областях науки и техники. Этим объясняется популярность данного пакета как в университетах и институтах, так и в научных лабораториях.

И наконец, важным достоинством пакета является возможность задания в блоках произвольных математических выражений, что позволяет решать типовые задачи, пользуясь примерами пакета Simulink или же просто задавая новые выражения, описывающие работу моделируемых пользователем систем и устройств. Важным свойством пакета является и возможность задания системных функций (S-функций) с включением их в состав библиотек Simulink. Необ-

ходимо отметить также возможность моделирования устройств и систем в реальном масштабе времени.

Как программное средство Simulink — типичный представитель визуально-ориентированного языка программирования. На всех этапах работы, особенно при подготовке моделей систем, пользователь практически не имеет дела с обычным программированием. Программа автоматически генерируется в процессе ввода выбранных блоков компонентов, их соединений и задания параметров компонентов.

Важное достоинство Simulink — это интеграция не только с системой MATLAB, но и с рядом других пакетов расширения, что обеспечивает, по существу, неограниченные возможности применения Simulink для решения практически любых задач имитационного и событийного моделирования.

Общие возможности Simulink

Даже предшествующая версия пакета Simulink 3.1 обладала следующими важными возможностями:

- интегрированный браузер моделей (Windows 95/98/NT);
- возможность увеличения блок-схем (zooming);
- блок Scope, способный работать с несколькими портами;
- интегрированные возможности линейного анализа;
- графический интерфейс для описания свойств сигнала;
- интегрированный браузер библиотек (только Windows 95/98/NT);
- новые блоки Subsystem, Round Sum, Enhanced Mux, Bus Selector и Model Info;
- поддержка различных типов данных и их преобразований;
- поддержка комплексных чисел при работе с базовыми блоками и комплексно-вещественные преобразования;
- оптимизация скорости и использования памяти при моделировании;
- многоцветные изображения, метки портов и маскированных блоков;
- блоки, определяемые пользователем, с поддержкой множества портов и различными интервалами дискретизации.

Дополнительные возможности версии Simulink 4.0

В новой версии Simulink 4.0 добавился еще ряд возможностей. Ниже они перечислены по категориям.

- Совершенствование пользовательского интерфейса:
  - Новый графический отладчик для интерактивного поиска и диагностики ошибок в модели.
  - Усовершенствован навигатор (браузер) моделей (Model Browser).
  - Новый однооконный режим для открытия подсистем.
  - Контекстное меню для блок-диаграмм (открывается щелчком правой кнопки мыши) как для версий Windows, так и для Unix.
  - Новое диалоговое окно Finder для поиска моделей и библиотек.
- Расширенная поддержка крупных приложений:
  - С помощью новых Simulink-объектов данных можно создавать специфические для приложений типы данных MATLAB.
  - Новый графический пользовательский интерфейс Simulink Explorer для наблюдения и редактирования объектов данных Simulink.
  - Усовершенствование блока Configurable Subsystems.
  - Новое меню выбора блока конфигурируемой подсистемы.
  - Поддержка защиты интеллектуальной собственности с помощью S-функций (требуется Real-Time Workshop 4.0).
  - Поддержка S-функций, кодируемых в языке ADA.
- Новые и улучшенные возможности блоков:
  - Поддержка матричных сигналов многими Simulink-блоками.
  - Блоки Product, Multiplication, Gain и Math Function теперь поддерживают матричные операции на матричных сигналах.
  - Блоки Mux и Demux теперь поддерживают мультиплексирование матричных сигналов.
  - Новый блок Reshape изменяет размерность своего входного сигнала.
  - Блок Probe теперь по умолчанию выводит размерность сигнала, подаваемого на вход.

- Новый блок Bitwise Logical Operator маскирует, обращает или расщепляет биты целочисленного сигнала без знака.
- Четыре новых блока Look-Up Table.
- Новый блок Polynomial выводит полиномиальную функцию от входного сигнала.

Все это говорит о том, что двенадцатая реализация системы (MATLAB 6.0 + Simulink 4.0) подверглась не «косметической», а самой серьезной переработке, выдвигающей эту систему на новый уровень развития и применения.

Запуск Simulink и основы работы с пакетом

Интеграция пакета Simulink с системой MATLAB

После инсталляции Simulink (отдельно от MATLAB или в его составе) он автоматически интегрируется с MATLAB. Внешне это выражается появлением кнопки Simulink в панели инструментов (перед кнопкой ?) системы MATLAB (см. рис. 1.4). При нажатии этой кнопки открывается окно интегрированного браузера библиотек, показанное в левой части рис. 2.1.

Нетрудно заметить, что пользовательский интерфейс окна браузера выполнен в общем стиле, характерном для Проводника Windows 95/98. Это позволяет отказаться от детального описания его особенностей. Отметим лишь главные возможности работы с браузером.

В окне браузера содержится дерево компонентов библиотек Simulink. Для просмотра того или иного раздела библиотеки достаточно выделить его мышью — в правой части окна Simulink Browser Library появится набор пиктограмм компонентов активного раздела библиотеки. На рис. 2.1 показан основной раздел библиотеки Simulink.

С помощью меню браузера или кнопок его панели инструментов можно открыть окно для создания новой модели или загрузить существующую. Работа с Simulink происходит на фоне открытого окна системы MATLAB, в котором нередко можно наблюдать за выполняемыми операциями — если их вывод предусмотрен программой моделирования.

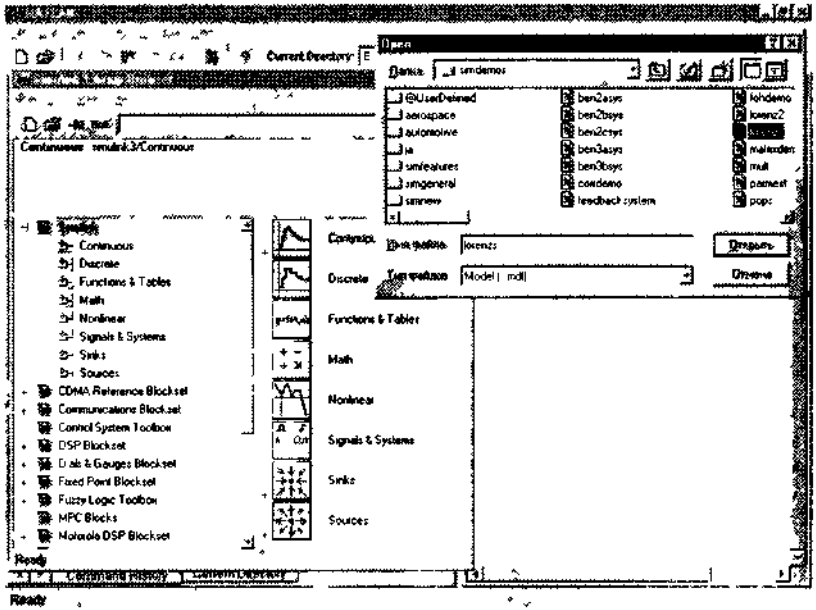


Рис. 2.1. Окно браузера библиотек Simulink

Интеграция пакета Simulink с системой MATLAB имеет глубокий смысл Решение большинства задач моделирования базируется на автоматическом составлении сложных конечно-разностных систем для линейных, нелинейных и дифференциальных уравнений, называемых уравнениями состояния модели Все это обеспечивает пакет Simulink.

Наиболее эффективное решение таких систем уравнений достигается за счет применения аппарата матричных вычислений, реализованного в системе MATLAB Вот почему сочетание Simulink с MATLAB оказалось столь плодотворным К этому стоит добавить, что Simulink использует практически любые операторы и функции системы MATLAB (а также язык программирования MATLAB)

Запуск моделей Simulink из среды MATLAB

Обычно Simulink запускается соответствующей кнопкой из панели инструментов, что было описано выше, после чего все последующие

действия выполняются в среде MATLAB + Simulink. Можно также запустить Simulink, выполнив в командной строке MATLAB команду

```
» simulink
```

Для вывода полного перечня команд Simulink надо выполнить команду

```
» help simulink
```

Список команд будет дан в следующей форме:

```
» help simulink
Simulink
Version 4.0 (R12) 16-Jun-2000
Model analysis and construction functions
Simulation
  Sim                 -   Simulate a Simulink model
  Sldebug             -   Debug a Simulink model
  Simset              -   Define options to SIM Options
                          structure
  Simget              -   Get SIM Options structure
                          Linearization and trimming
  Linmod              -   Extract linear model from
                          continuous time system
  linmod2             -   Extract linear model  advanced
                          method
  Dlinmod             -   Extract linear model from discrete-
                          time system
  Trim                -   Find steady-state operating point
  Model Construction
  close_system        -   Close open model or block
  new_system          -   Create new empty model window
  open_system         -   Open existing model or block
  load_system         -   Load existing model without making
                          model visible
  save_syste          -   Save an open model
  add_block           -   Add new block
  add_line            -   Add new line
  delete_block        -   Remove block
  delete_line         -   Remove line
  find_system         -   Search a model
  hilite_system       -   Hilite objects within a model
  replace_block       -   Replace existing blocks with a new
                          block
  set_param           -   Set parameter values for model or
                          block
  get_param           -   Get simulation parameter values from
                          model
  Bdclose             -   Close a Simulink window
  Bdroot              -   Root level model name
  Gcb                 -   Get the name of the current block
```

```
Gcbh            -   Get the handle of the current block
Gcs                 -   Get the name of the current system
Getfullname     -   get the full path name of a block
Slupdate        -   Update older 1 x models to 3 x
Addterms        -   Add terminators to unconnected
                        ports
Boolean         -   Convert numeric array to boolean
Slhelp              -   Simulink user's guide or block help
Masking
Hasmask         -   Check for mask
Hasmaskdlg          -   Check for mask dialog
Hasmaskicon     -   Check for mask icon
Iconedit        -   Design block icons using ginput
                        function
Maskpopups          -   Return and change masked block's
                        popup menu items
Movemask        -   Restructure masked built-in blocks
                        as masked subsystems
Library
Libinfo         -   Get library information for a
                        system
Diagnostics
Sllastdiagnostic    -   Last diagnostic array
Sllasterror     -   Last error array
Sllastwarning   -   Last warning array
Sldiagnostics   -   Get block count and compile stats
                        for a model
Hardcopy and printing
Frameedit       -   Edit print frames for annotated
                        model printouts
Print               -   Print graph or Simulink system, or
                        save graph to M-file
Printopt        -   Printer defaults
Orient              -   Set paper orientation
See also BLOCKS and SIMDEMOS
```

Дополнительную информацию можно получить, используя команды `help blocks` и `help simdemos`. Первая команда дает информацию об основных библиотеках Simulink и примерах применения S-функций, а вторая выводит список демонстрационных примеров. Запуск этих примеров дает возможность практически познакомиться с возможностями пакета Simulink и оценить степень сложности систем и устройств, которые могут моделироваться с помощью этого пакета.

Особенности интерфейса Simulink 4.0

Интерфейс версии Simulink 4.0 полностью соответствует стилю интерфейса *типичных приложений Windows 95/98/NT/2000*, в том

числе интерфейсу системы MATLAB. В то же время он концептуально строг, чтобы не досаждать пользователю многочисленными «излишествами» стандартного интерфейса Windows 95/98/NT/2000. Меню системы содержит следующие пункты.

- File — работа с файлами моделей и библиотек (их создание, сохранение, считывание и печать).
- Edit — операции редактирования, работа с буфером обмена и создание подсистем.
- View — управление отображением панели инструментов и строки состояния.
- Simulation — управление процессом моделирования (старт, пауза, вывод окна настройки параметров моделирования).
- Format — операции форматирования модели (смена шрифтов, редактирование надписей, повороты блоков, использование тени от блоков, операции с цветами линий блоков, их фоном и общим фоном.
- Tools — управление видом анализа (в линейной области и в режиме реального времени RTW).

Первые три пункта главного меню содержат общепринятые для Windows-приложений команды и операции, поэтому мы не будем их обсуждать. Остальные будут рассмотрены по мере знакомства с системой Simulink.

Как уже отмечалось, вместе с рабочим окном Simulink выводится окно с перечнем разделов основной библиотеки компонентов. Это окно — важная часть интерфейса Simulink. Оно открывает доступ к другим пакетам компонентов (Blocksets&Toolboxes) и примерам их применения (Demos). Это дает пользователю возможность постепенно знакомиться с новыми областями применения Simulink.

Примеры моделирования систем

Моделирование аттрактора Лоренца

Поиск и загрузка примера моделирования

Даже не знакомый с Simulink пользователь может быстро оценить возможности этого пакета, воспользовавшись интересными и поучительными примерами, входящими в поставку Simulink. Они нахо-

дятся в папке MATLAB/TOOLBOX/SIMULINK/SIMDEMOS. Попутно отметим, что в папке MATLAB/SIMULINK располагаются служебные файлы.

Для загрузки примеров используется (как обычно) команда Open в меню File браузера библиотеки Simulink. В окне с названием Open (рис. 2.1) надо войти в указанную папку и выбрать подходящий пример. Окно Open можно растянуть для получения более полного обзора файлов демонстрационных примерв (рис. 2.2)

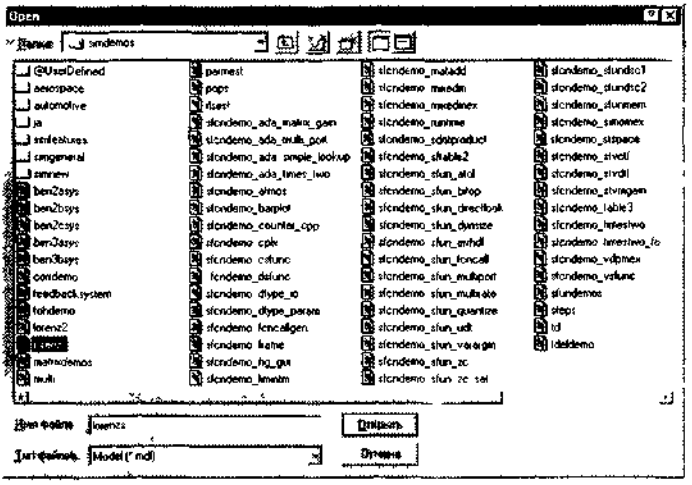


Рис. 2.2. Окно с полным перечнем файлов демонстрационных примеров пакета Simulink

При открытии нужного примера (в данном случае — файл lorenzs) появляется окно редактирования графической модели устройства (рис. 2.3).

Как можно заметить из примера, графическая модель содержит ряд блоков. Каждый блок имеет наглядное общепринятое обозначение в виде прямоугольника, треугольника и т. д. Блоки имеют входы и выходы и описываются различными математическими зависимостями. Блоки соединяются друг с другом линиями со стрелками, причем стрелка указывает направление от выходов одних блоков ко входам других. Имеются также текстовые комментарии и средства для вывода подсказок и открытия окон справочной системы.

Аттрактор Лоренца является типичной замкнутой колебательной системой с положительной обратной связью, в которой развиваются колебания очень сложной формы, демонстрирующие явление

хаоса. Модель содержит ряд вычислительных модулей, отражающих ее нелинейность и основные особенности структуры, а также два регистрирующих прибора — виртуальные осциллограф и графопостроитель.

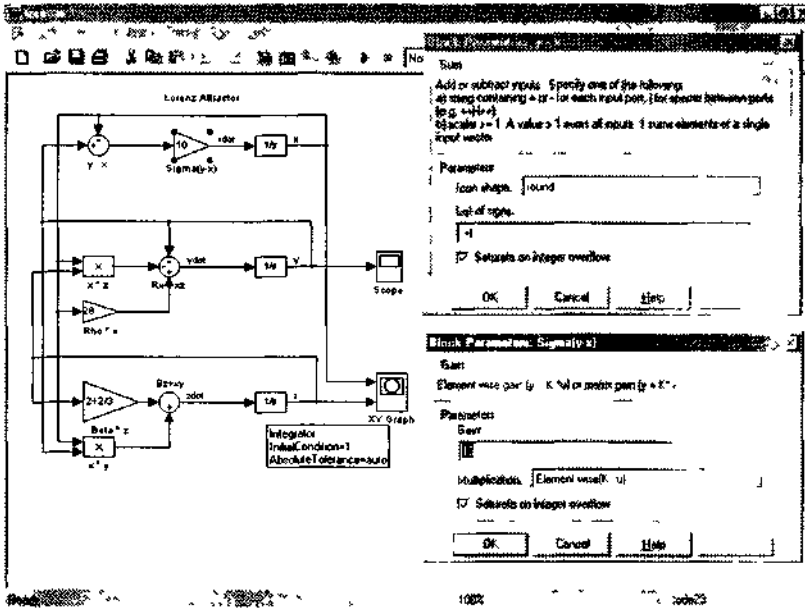


Рис. 2.3. Модель аттрактора Лоренца в окне редактирования пакета Simulink

Как устанавливаются параметры компонентов модели?

Для пользователя, имеющего хотя бы начальное представление о моделировании, вполне ясно, что любой компонент модели имеет какие-то параметры или хотя бы один параметр. Из самой графической модели не видно, какими параметрами обладает тот или иной компонент. Каким же образом узнать, что за параметры компонентов установлены в той или иной модели?

Для того чтобы вызвать окно модификации параметров компонента, нужно навести курсор мыши на изображение компонента и дважды щелкнуть левой кнопкой мыши. На рис. 2.3 справа от модели показаны два таких окна для компонентов с именами y-x и Sigma(y-x).

Кроме того, можно просто просмотреть параметры компонента. Для этого нужно, указав курсором мыши интересующий вас компонент, задержать курсор на 2–3 секунды. Под ним появится подсказка с указанием параметров этого компонента — на рис. 2.3 показана подсказка для компонента 1/s.

Установка параметров моделирования

Прежде чем запустить загруженную модель, стоит ознакомиться с установкой общих параметров моделирования. Для этого выполним команду Simulation Parameters... в меню Simulation окна Simulink. Появится окно установки параметров моделирования, показанное на рис. 2.4.

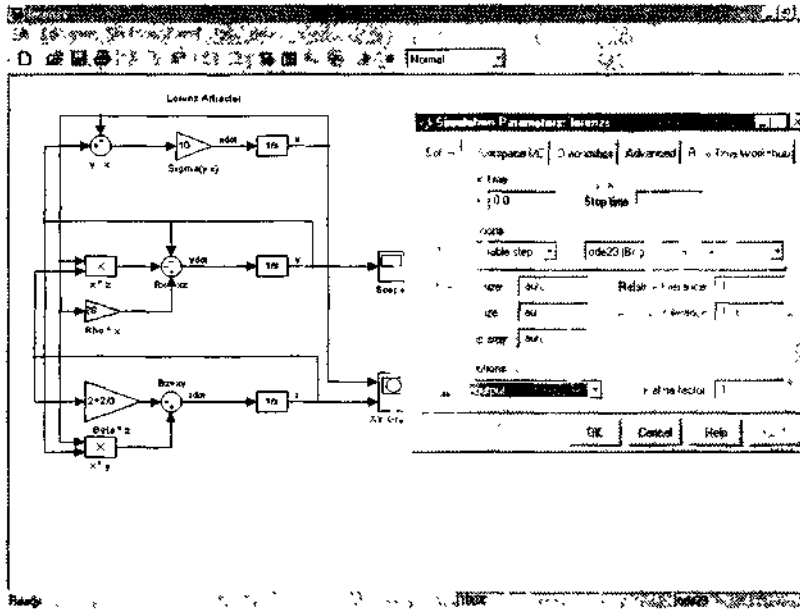


Рис. 2.4. Установка параметров модели аттрактора Лоренца

Это окно имеет ряд вкладок с довольно большим числом параметров. Мы постепенно рассмотрим их все. Но пока остановимся на вкладке, открытой по умолчанию, — Solver (Решатель). Эта вкладка позволяет установить параметры решающего устройства системы моделирования Simulink.

К числу важнейших параметров решателя относится время моделирования — Simulation time. Оно задается начальным временем Start

time (обычно 0) и конечным временем Stop time (в нашем случае бесконечность inf). Равенство Stop time бесконечности означает, что моделирование будет происходить бесконечно долго, пока мы не прервем его. Однако в этом случае трудно получить различимые осциллограммы работы устройства, поэтому рекомендуется задавать конечные значения Stop time.

Стоит отметить, что время моделирования — величина довольно условная. Не следует думать, что Stop time=50 означает моделирование в течение 50 секунд. Точного соответствия между временем моделирования в секундах и заданным значением нет. Реальное время моделирования сильно зависит от быстродействия компьютера, на котором выполняется моделирование.

Первостепенное значение имеют две опции решателя в поле Solver options: тип решения и метод решения. Возможны два типа решения:

- Variable-step solvers — решение с переменным шагом.
- Fixed-step solvers — решение с фиксированным шагом.

Как правило, лучшие результаты дает решение с переменным шагом (обычно по времени, но не всегда). В этом случае шаг автоматически уменьшается, если скорость изменения результатов в процессе решения возрастает. И напротив, если результаты меняются слабо, шаг решения автоматически увеличивается. Это исключает (опять-таки, как правило) расхождение решения, которое нередко случается при фиксированном шаге.

Метод с фиксированным шагом стоит применять только тогда, когда фиксированный шаг обусловлен спецификой решения задачи, например, если ее цель заключается в получении таблицы результатов с фиксированным шагом. Этот метод дает неплохие результаты, если поведение системы описывается почти монотонными функциями.

Вторая из указанных опций — выбор метода моделирования. Рисунок 2.5 показывает окно установки параметров моделирования отдельно с открытым меню выбора метода моделирования.

Для решения дифференциальных уравнений можно выбрать следующие методы: discrete (дискретный), ode45, ode23 (три варианта, включая метод Розенброка), rk45 (метод Дорманда—Принса), ode113 (метод Адамса) и ode15s. Методы, в наименовании которых имеется слово stiff, служат для решения жестких систем дифференциальных уравнений.

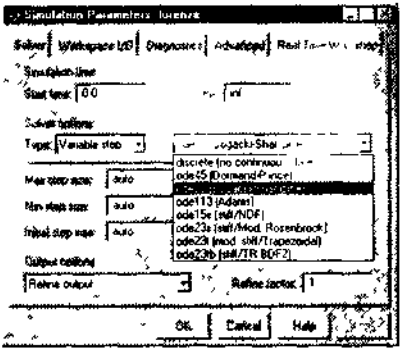


Рис. 2.5. Установка метода моделирования

(Подробное описание указанных методов решения дифференциальных уравнений можно найти в книгах по системе MATLAB, рекомедованных в главе 1.)

ВНИМАНИЕ

Выбор метода моделирования имеет решающее значение в достижении успеха моделирования. В этом отношении демонстрационные примеры Simulink, будучи заранее тщательно отлаженными, играют злую шутку с начинающими пользователями. У последних складывается совершенно неверное представление о легкости моделирования. На самом деле подготовка модели даже не очень сложной системы требует серьезных знаний и немалых трудов по ее отладке.

Следующие три параметра — значения опции auto — обычно задаются автоматически, но в особых случаях их можно ввести явно.

- Max step size — максимальный шаг интегрирования системы однородных дифференциальных уравнений;
- Min step size — минимальный шаг интегрирования;
- Initial step size — начальный шаг интегрирования.

Важен и такой параметр моделирования, как точность интегрирования:

- Relative tolerance — относительная погрешность интегрирования;
- Absolute tolerance — абсолютная погрешность интегрирования.

По умолчанию они заданы соответственно равными 10^{-3} и 10^{-6}. Если, например, графики результатов моделирования выглядят составленными явно из отдельных фрагментов, это указывает на необходимость уменьшения указанных значений погрешности. Однако слиш-

ком малые погрешности могут вызвать значительное увеличение времени вычислений. Не оптимально выбранные значения погрешности (как очень малые, так и очень большие) могут вызвать неустойчивость и даже «зацикливание» процесса моделирования.

С остальными параметрами и вкладками окна параметров моделирования мы познакомимся в дальнейшем.

Запуск процесса моделирования

В конце панели инструментов Simulink находятся две важные кнопки управления моделированием. Одна из них, в виде черного треугольника (Start/Pause Simulation), запускает или приостанавливает начатый процесс моделирования, а другая, в виде черного квадратика (Stop), останавливает его. Все, что нужно для запуска моделирования, — это нажать кнопку с изображением треугольника. Рисунок 2.6 показывает результат запуска выбранной модели. Вместо кнопок можно использовать команды Start и Pause в меню Simulation окна модели.

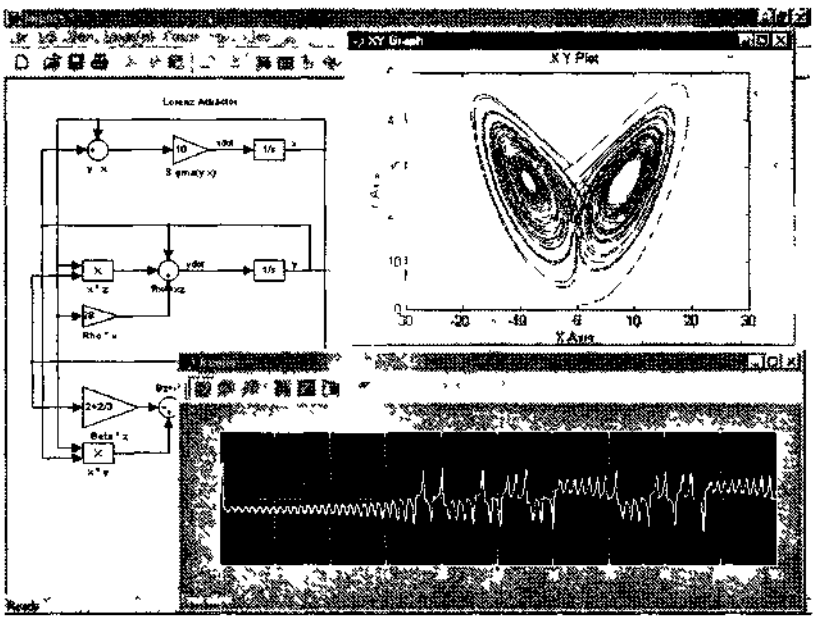


Рис. 2.6. Результаты моделирования аттрактора Лоренца

В данном случае результаты моделирования представлены в виде довольно сложного и неординарного фазового портрета колебаний,

построенного с помощью виртуального графопостроителя, и осциллограммы временной зависимости колебаний, полученной с помощью виртуального осциллографа.

Результат моделирования показывает, что даже в такой сравнительно простой нелинейной системе, каковой является аттрактор Лоренца, возникают сложные и отчасти хаотические колебания.

Решение дифференциальных уравнений Ван-дер-Поля

Пример системы Ван-дер-Поля

Продолжим рассмотрение моделирования колебательных систем.

Системы второго порядка, описываемые дифференциальными уравнениями второго порядка, занимают важное место в таком фундаментальном разделе физики, как теория колебаний. Именно на базе таких систем созданы многочисленные генераторы периодических колебаний самого различного типа — от LC-генераторов до лазерных и иных квантовых генераторов.

Пример моделирования типовой системы второго порядка есть в наборе демонстрационных примеров пакета Simulink. Модель такой системы представлена на рис. 2.7. Эта система описывается хорошо известным нелинейным дифференциальным уравнением второго порядка Ван-дер-Поля (или двумя уравнениями первого порядка).

Как нетрудно заметить, данная модель представляет собой усилитель с нелинейным элементом Fcn, позволяющим задать тип нелинейности, с положительной обратной связью и имеет в своем тракте блоки, ослабляющие как высокие, так и низкие частоты. Колебания в такой системе возникают на некоторой частоте, для которой фазовый сдвиг тракта равен нулю, а малосигнальный петлевой коэффициент передачи превышает 1. Характер развития колебательного процесса в решающей мере зависит от характера нелинейности, заданного в блоке Fcn.

На рис. 2.7 показан и результат моделирования. В данном случае виртуальный осциллограф отображает две кривые, соответствующие выходным портам Out 1 и Out 2. Для этого перед осциллографом размещен блок мультиплексора Mux с двумя входами. Результат моделирования отображается в виде временных зависимостей выходных сигналов.

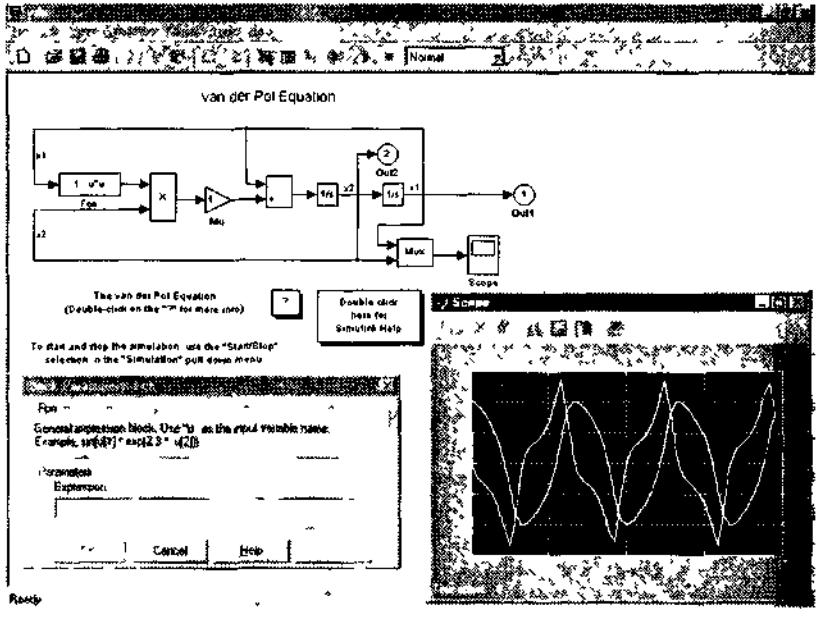


Рис. 2.7. Модель колебательной системы второго порядка

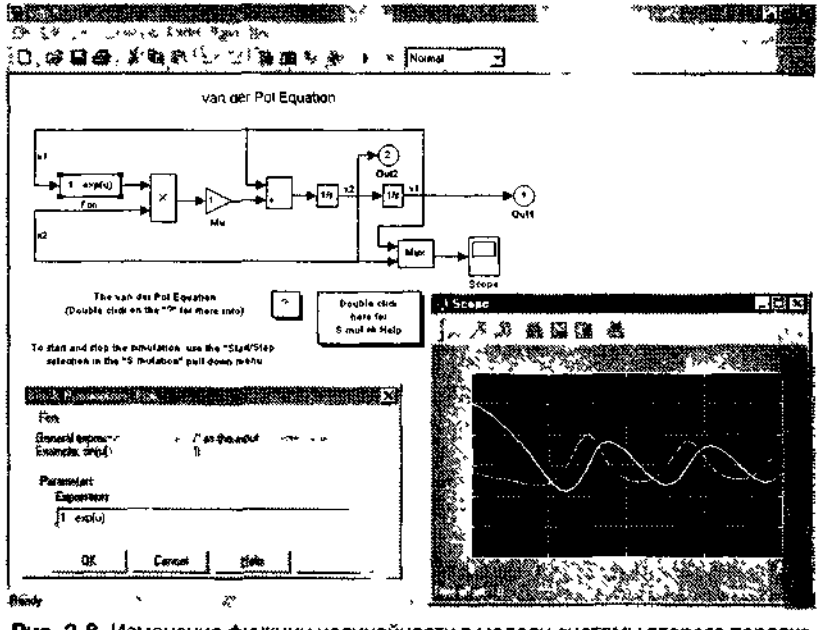


Рис. 2.8. Изменение функции нелинейности в модели системы второго порядка

Как изменить характер нелинейности

Результат на рис. 2.7 получен для зависимости вида $F(u) = 1 - u*u$ (окно задания нелинейности представлено на рис. 2 7 в левом нижнем углу). Допустим, нас интересует поведение системы для иной нелинейности, скажем $F(u) = 1 - \exp(u)$ Для замены нелинейности достаточно сделать двойной щелчок на блоке Fcn. В появившемся окне параметров надо вместо функции по умолчанию ввести новую функцию, отражающую нелинейность модели. Это и показано на рис. 2.8.

После уточнения нелинейности и закрытия окна параметров можно запустить измененную модель. Результаты также представлены на рис. 2.8. Сравнение временных диаграмм (осциллограмм) выходных сигналов на рис. 2.7 и 2.8 показывает существенные изменения в характере поведения системы. Во втором варианте предварительная стадия занимает больше времени и колебания во время переходного процесса имеют существенно большую амплитуду, чем в стационарном режиме.

Как добавить в модель графопостроитель

Иногда поведение системы второго порядка удобно представить фазовым портретом колебаний — как в нашем первом примере (рис. 2 6). Но во втором примере вывод фазового портрета не предусмотрен. Добавить его очень просто — надо отредактировать имеющуюся модель.

Эту модель достаточно дополнить графопостроителем, входы которого подключаются к выходным портам Out 1 и Out 2. Для этого сначала выведем на передний план окно браузера библиотек Simulink и откроем в нем раздел Sinks (Регистрирующие устройства). Найдя в нем компонент XY-Graph, перенесем его в окно модели и расположим справа от осциллографа. Этот момент работы поясняет рис. 2.9.

Теперь надо подключить входы графопостроителя к выходным портам. Для этого, нажав клавишу Ctrl и удерживая ее, уцепитесь курсором мыши за провод, подходящий к порту 1. Начните перемещать курсор мыши к верхнему входу графопостроителя при нажатой левой кнопке мыши. Вслед за курсором мыши будет тянуться создаваемое соединение. Указав вход графопостроителя, отпустите кнопку мыши и клавишу Ctrl Первое соединение сделано. Если оно пересекает какой-то блок, то, захватив линию соединения и нажав левую кнопку мыши, отведите соединение в нужное место.

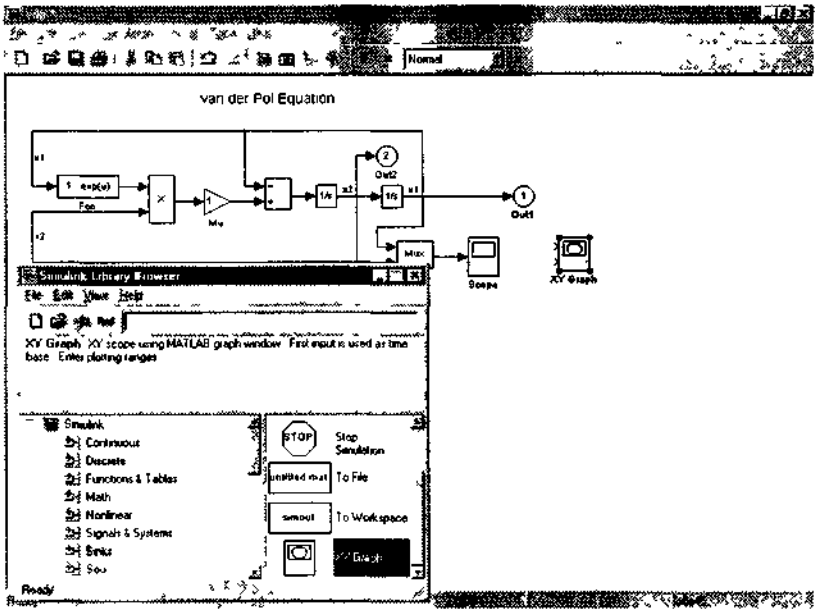


Рис. 2.9. Перенос графопостроителя в модель

Аналогичным образом соедините порт 2 с нижним входом графопостроителя. Модель примет вид, показанный на рис. 2.10.

Теперь скорректируем нелинейность: $F(u) = 1 - 1.1*\exp(-u)$. В данном случае параметры нелинейности подобраны таким образом, что колебательный процесс возникает только в начале включения системы. Затем за несколько периодов колебания затухают. Запустив модель, нетрудно убедиться в этом: результаты моделирования на рис. 2.10 наглядно показывают, что фазовый портрет колебаний — это сворачивающаяся спираль, а временные зависимости — затухающие во времени колебания.

В этих примерах мы столкнулись с принципиально важным достоинством пакета Simulink — аналитическое описание многих моделей можно менять, причем оно выполняется по правилам, принятым в системе MATLAB. Благодаря этому математическая сущность модели оказывается вполне понятной, а результаты моделирования наглядно и адекватно описывают работу сложных моделей при введении в их описание самых разных математических закономерностей.

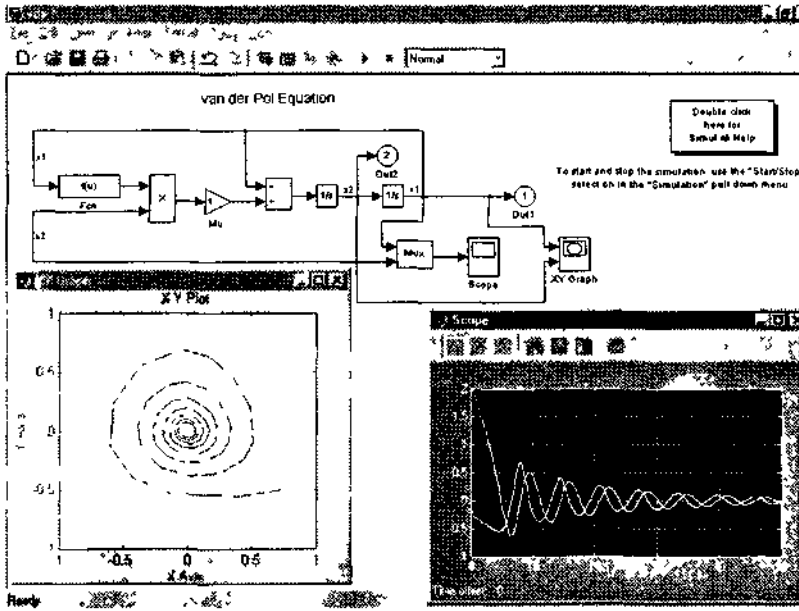


Рис. 2.10. Модель с подключенным графопостроителем

Работа с редактором дифференциальных уравнений

Решение уравнений Ван-дер-Поля

Приведенный выше пример является характерным для ситуации, когда моделируется система или устройство, поведение которого описывается дифференциальными уравнениями известного вида — в нашем случае уравнениями Ван-дер-Поля. Однако Simulink имеет специальный редактор дифференциальных уравнений, с помощью которого можно задать систему дифференциальных уравнений первого порядка явно в общепринятой форме Коши и тут же начать ее решение с помощью решателя. Для получения доступа к решателю надо загрузить файл dee, который находится в папке MATLAB/TOOLBOX/SIMULINK/DEE.

Ограничимся примером с именем ddedemo1. Он выводит окно всего с двумя блоками: блоком vdp и осциллографом x1. Первый блок решает заданное уравнение Ван-дер-Поля, а второй просто отобража-

ет решение в виде временной зависимости. Двойной щелчок на блоке vdp открывает редактор дифференциальных уравнений, окно которого показано на рис 2.11 в правой части. В этом окне можно модифицировать решаемые уравнения или ввести новые. Окно осциллографа также представлено на рис. 2.11.

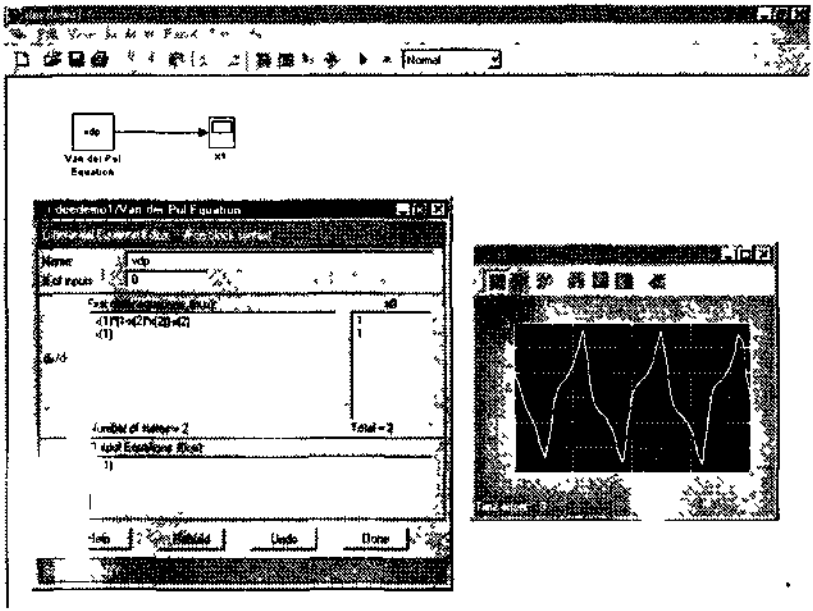


Рис. 2.11. Решение систмы дифференциальных уравнений Ван-дер-Поля

Решение уравнений аттрактора Лоренца

Теперь вернемся к уже описанному аттрактору Лоренца. Файл deedemo2 дает пример задания системы дифференциальных уравнений для аттрактора Лоренца в явном виде и их решения (рис. 2.12).

Нетрудно заметить, что результат решения совпадает с результатом моделирования, представленным на рис. 2.6.

В системе MATLAB одну и ту же задачу можно решать рядом способов. В этом случае получение одинаковых результатов (в том числе при решении средствами Simulink) дает дополнительную гарантию правильности решения и корректности создаваемых моделей.

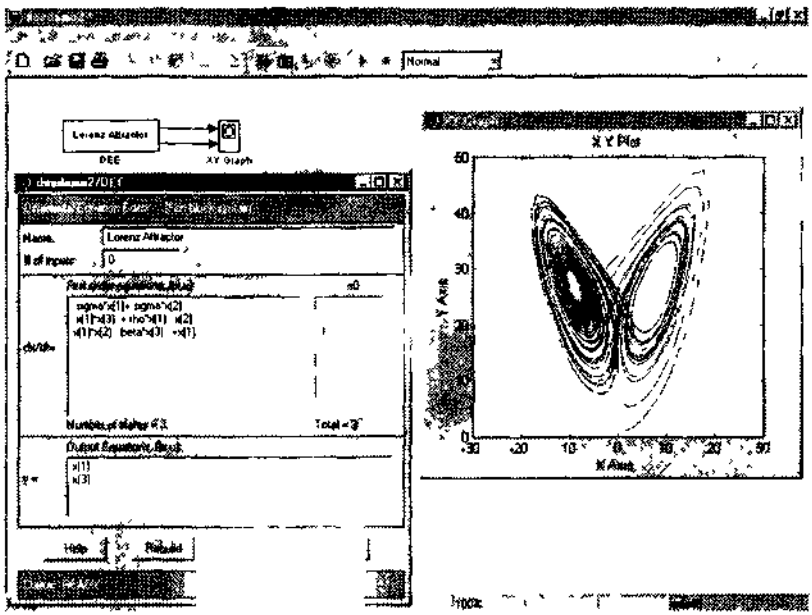


Рис. 2.12. Решение системы дифференциальных уравнений аттрактора Лоренца

Моделирование кубика с пружинкой

Модель кубика

Теперь рассмотрим следующий пример из области физики.

Пусть кубик, сбоку которого прикреплена пружинка, лежит на *гладкой поверхности. Как будет вести себя кубик, если мы,* ухватившись за свободный конец пружинки, будем дергать ее туда-сюда? Знающий физику человек сразу скажет, что если трение не очень велико (а у нас поверхность гладкая), то кубик будет перемещаться из одного положения в другое с затухающими колебаниями.

Затухание колебаний обусловлено демпфированием вследствие трения кубика о поверхность, на которой он лежит. Это, кстати, напоминает поведение аттрактора Лоренца (на его фазовом портрете тоже видны два характерных состояния), но в данном случае перемещение кубика обходится без хаотических движений.

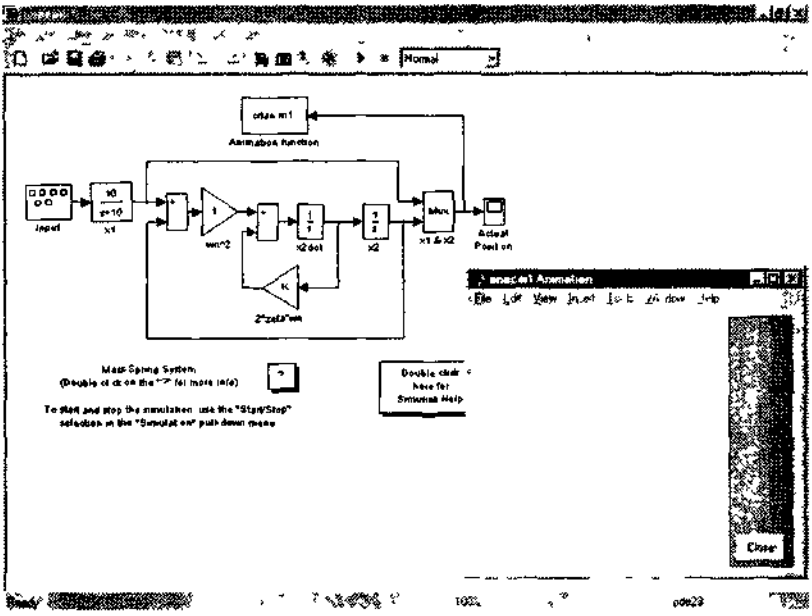


Рис. 2.13. Модель кубика с пружинкой

Файл этого примера – onecart – находится в папке simgeneral. Рисунок 2.13 показывает модель этого примера и результат моделирования.

В данном случае результат моделирования отображается в виде анимационного видеоклипа. Наглядность представления поведения устройства в данном случае вполне очевидна. К сожалению, рис. 2.13 показывает лишь один кадр анимации – на деле отчетливо видны рывки палочки, на которой закреплен свободный конец пружины, и затухающие колебания кубика.

Информационное обеспечение примера

Помимо самой модели в окне ее редактирования можно вводить текстовые комментарии (аннотации) и гиперссылки в виде прямоугольников разного размера с надписями. Две гиперссылки показаны под моделью на рис 2.13. При активизации гиперссылки выводится информационное окно (рис. 2.14).

В остальном работа с этой моделью и коррекция ее параметров аналогичны описанной выше.

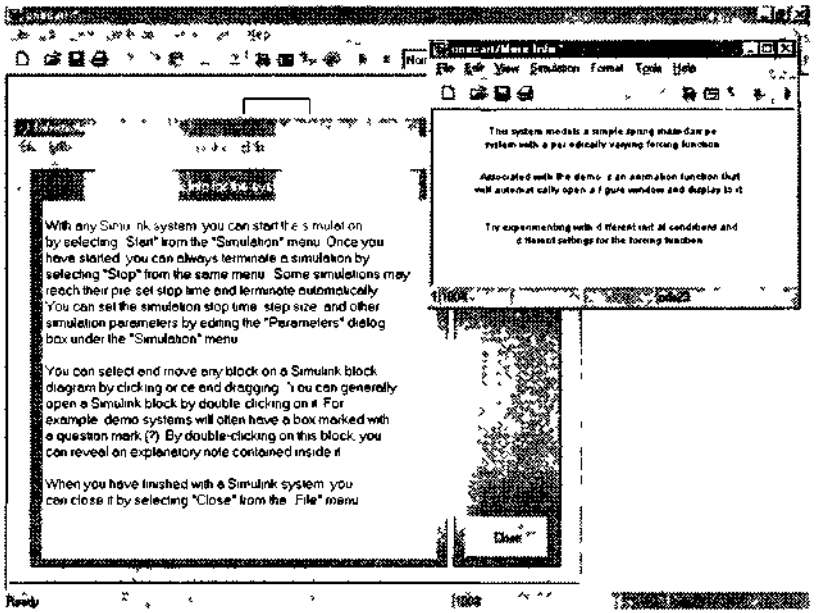


Рис. 2.14. Активизация гиперссылок

Моделирование системы терморегулирования дома

Рисунок 2.15 показывает модель системы терморегулирования для отдельного дома.

Система терморегулирования представляет собой замкнутую релаксационную систему. Температура в доме контролируется с помощью датчика температуры. Его сигнал сравнивается с опорным сигналом температуры, и разность используется для импульсного управления нагревателем. В системе предусмотрен контроль расходов на обогрев дома (в долларах). Осциллограммы работы модели, полученные с помощью виртуального осциллографа, представлены на рис. 2.15.

Использование субмоделей

В этом примере мы сталкиваемся с новой принципиально важной и эффективной возможностью пакета Simulink — использованием субмоделей. Субмодель строится практически так же, как и модель сис-

темы. Она имеет обозначенные цифрами порты входа и выхода, через которые соединяется с основной моделью.

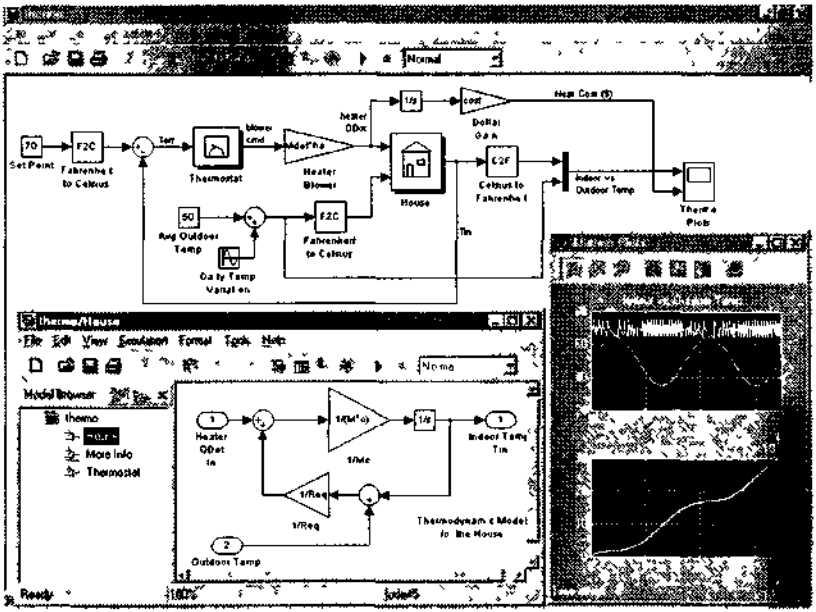


Рис. 2.15. Система терморегулирования дома

На рис. 2.15 показана одна из новинок версии Simulink 4 0 — браузер субмоделей. Он находится внизу экрана. В левом окне браузера расположена древовидная файловая структура субмоделей, а в правой — выделенная субмодель, в нашем случае это термодинамическая модель дома.

На рис. 2.16 представлена другая субмодель — субмодель термостата. Она представляет собой типичное релейное устройство, срабатывающее, если температура достигает заданного значения.

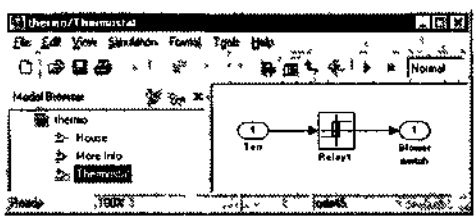


Рис. 2.16. Субмодель термостата

Моделирование работы унитаза

Модель унитаза

Для тех, кому приведенные выше примеры кажутся слишком далекими от обыденной жизни, приведем еще один пример — моделирование работы столь известного устройства первейшей необходимости, как унитаз. Модель сливного бачка унитаза представлена на рис. 2.17.

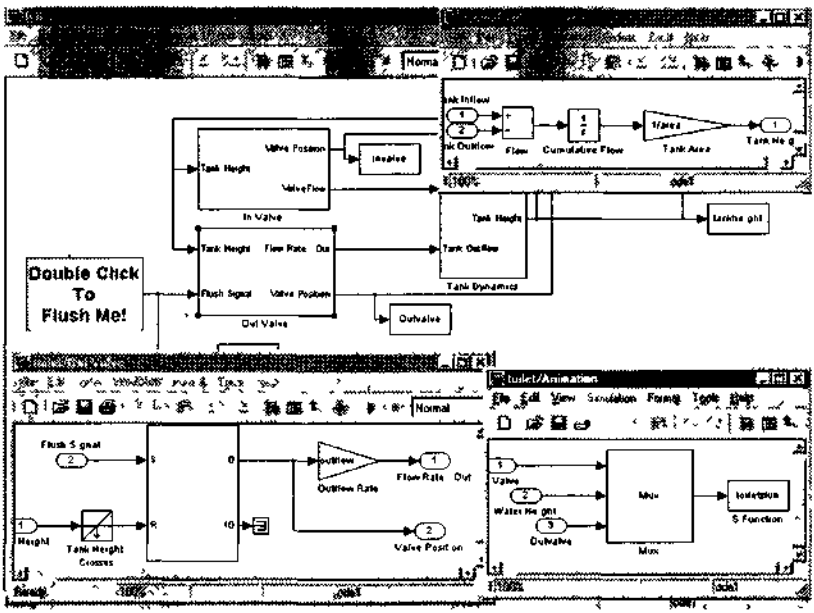


Рис. 2.17. Модель сливной системы унитаза

В этом примере моделируется процесс заполнения водой сливного бачка унитаза с ограничением уровня заполнения, а затем процесс слива воды при открывании соответствующего клапана. Слив сопровождается анимационным видеоклипом и звуковым комментарием, буквально воспроизводящим звуки слива воды в унитазе.

Субмодели унитаза

Несмотря на то что унитаз кажется довольно примитивным устройством по сравнению, скажем, с самолетом или космическим кораблем, его модель не так уж и проста. Как видно из рис. 2.17, в модель

унитаза входит ряд субмоделей — на этот раз они показаны не в среде браузера подмоделей, а в разных окнах. Так было принято в более ранних версиях системы Simulink.

Этот пример, как и предшествующий, дает хорошее представление о технике применения субмоделей. Деление на субмодели позволяет создавать основную модель более простого и более наглядного вида, чем в том случае, когда субмодели не применяются.

Контроль за сливом воды в унитазе

Данный пример демонстрирует всю мощь Simulink в смысле контроля за виртуальной работой моделируемого устройства. Рисунок 2.18 показывает визуальные средства контроля для этого примера.

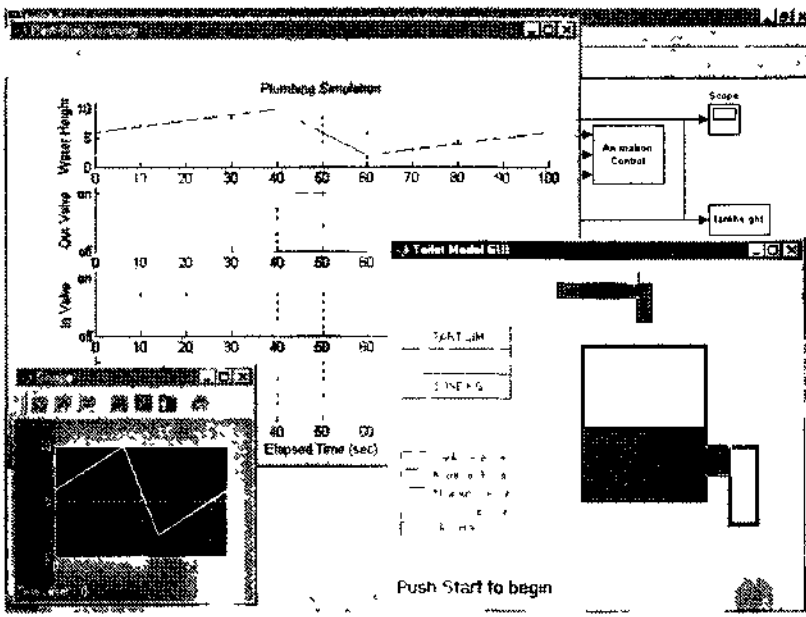


Рис. 2.18. Модель сливной системы унитаза

Уже говорилось, что слив воды в нашем виртуальном унитазе сопровождается воспроизведением звука. Разумеется, для прослушивания звука компьютер должен иметь звуковую карту и подключенные к ней акустические колонки. Словом, надо располагать мультимедийным компьютером. Это пригодится и для более серьезных задач по обработке звуковых сигналов в Simulink.

На рис. 2.18 видна целая гамма средств визуального контроля — это и осциллограмма уровня воды в баке, и детальные графические диаграммы процессов накопления и слива воды, и, наконец, окно с анимационным рисунком, показывающим, в какой момент какая задвижка открывается или закрывается и как происходит наполнение бака водой и ее слив в нужный момент.

Моделирование механических колебательных систем

Простой маятник

Динамические системы широко представлены в механике. Мы рассмотрим две такие системы. На рис. 2.19 представлена простая система механического маятника — подвешенный за верхний конец стержень.

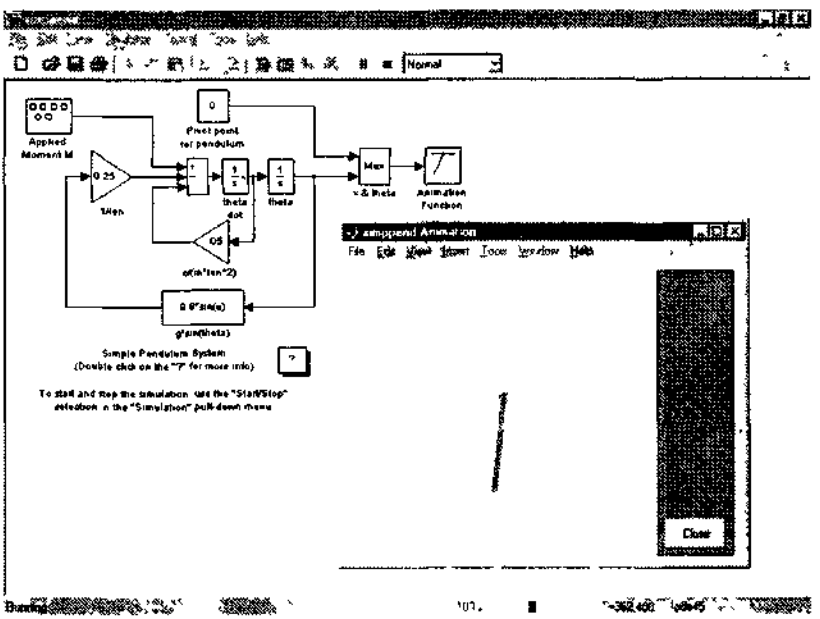


Рис. 2.19. Модель механического маятника

Блок Animation Function выводит графическое окно анимационного изображения колебаний. На нем можно видеть колебания стержня — маятника.

Колебания многозвенного объекта

Гораздо сложнее оказывается поведение систем, находящихся под внешним воздействием. Рисунок 2.20 показывает модель колебаний объекта, состоящего из трех подвижно сочлененных стержней, находящихся под сложным воздействием.

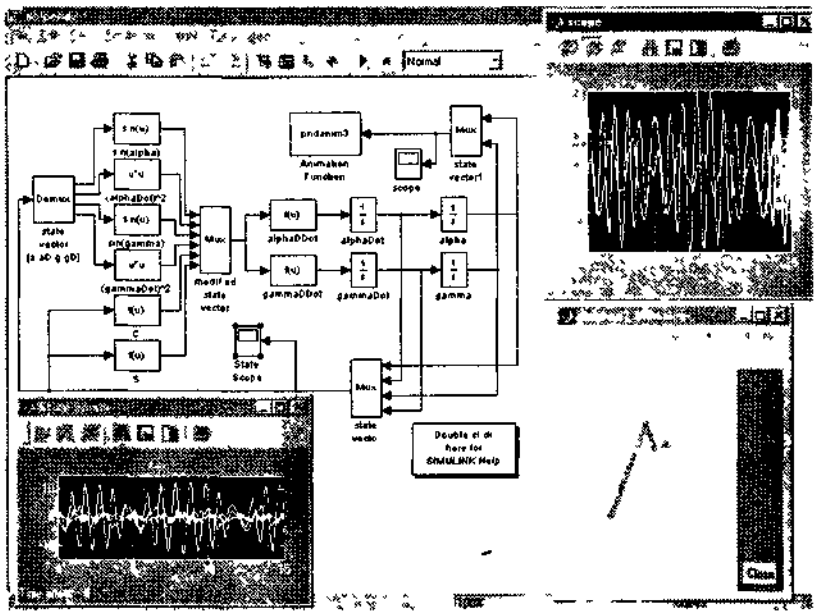


Рис. 2.20. Моделирование колебаний многозвенного объекта

Колебания такого объекта подчас могут носить весьма причудливый характер. Он иллюстрируется анимационными движениями многозвенного объекта.

Применение логических операций

Многие системы и устройства содержат блоки логических операций, которые подробно описывались в главе 1. Рисунок 2.21 показывает процесс моделирования системы, в которой блоки логических операций AND (И) и NOT (НЕ) используются для генерации трех последовательностей прямоугольных импульсов с частотами повторения, кратными 1, 2 и 3.

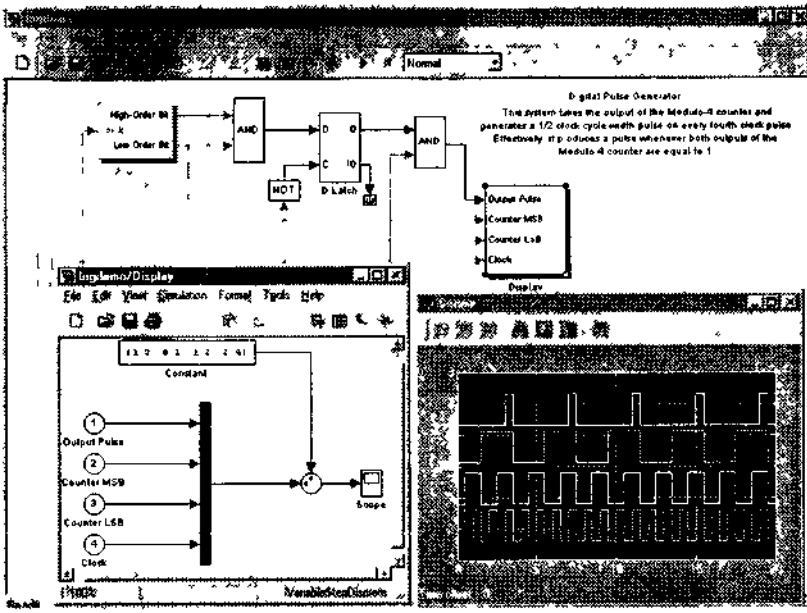


Рис. 2.21. Модель генератора трех последовательностей прямоугольных импульсов

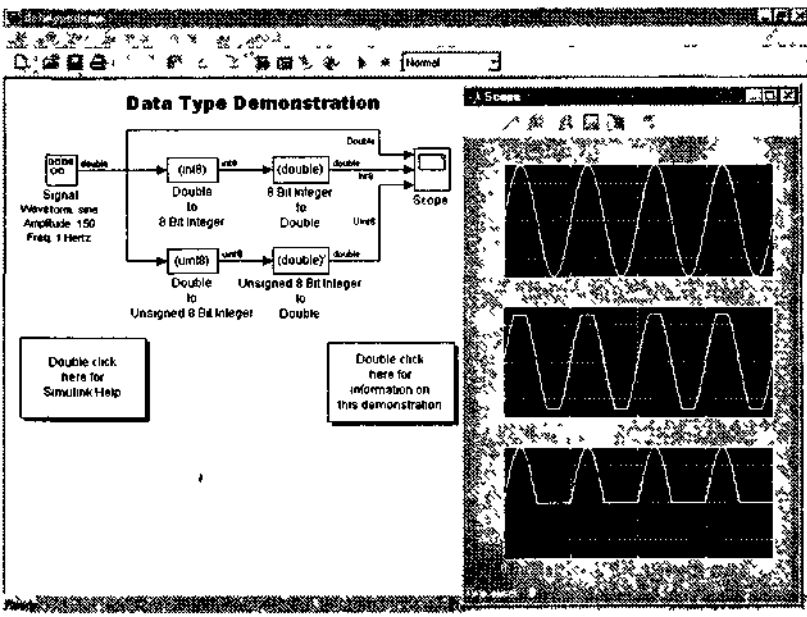


Рис. 2.22. Модель с указанием типов примененных в ней данных

Визуальный контроль типов данных

Simulink, как и MATLAB, работает с данными различного типа. Чаще других используются данные в формате чисел с плавающей точкой и двойной длиной — double. Могут использоваться и 8-разрядные целые числа двойной длины и прочие типы данных.

Форматы данных на входах и выходах блоков будут выводиться (рис. 2.22), если установить флажок Plot data type в меню Tools окна модели Simulink.

В большинстве случаев отображение типов данных лишь загромождает модель. Однако при отладке сложных моделей он может быть полезным.

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 себе нужные блоки модели, соединил их друг с другом, запустил в режиме моделирования — и почивай 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 при работе достаточно опытного пользователя, реально поработавшего с тем или иным пакетом расширения не один десяток часов и интуитивно чувствующего правоту (или неправоту) своих действий

Малоопытный пользователь, скорее всего, при переходе к моделированию своих систем или устройств 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. По-

этому и автор данной книги был вынужден ограничиться лишь некоторыми общими рекомендациями.

Довольно часто причиной ошибок является несоответствие типов блоков и их входных и выходных параметров. В таких случаях надо предусматривать переходные элементы. Наглядный пример — переход от тока к напряжению включением резистора 1 Ом в цепь тока.

Особенно часто нестыковка блоков наблюдается при совместном использовании блоков из разных пакетов расширения, например из пакетов Power System и Simulink. Размерные величины, 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.* в этом смысле не лишена серьезных недостатков. Довольно часто в ней наблюдается нестыковка моделей компонентов, взятых из разных пакетов расширения. В связи с этим стоит отметить, что многие из этих недостатков у новейшей версии MATLAB 6.0 с пакетами расширения устранены. Тем не менее и в MATLAB 6.0 подобные ситуации не так уж и редки.

По-видимому, стоит разумно ограничить применение 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 + Simulink — это уже «высший пилотаж», и он достигается только долгими тренировками и практикой.

Утешением для большинства пользователей MATLAB с пакетами расширений является то обстоятельство, что система тщательно диагностирует подготовленную модель и допускает ее исполнение только после устранения всех обнаруженных ошибок. Сообщения об ошибках появляются в специальных окнах системы. Они достаточно подробны и позволяют наметить меры по устранению ошибок.

Поэтому стоит отметить, что примеры, приведенные в книге и в справочной базе данных MATLAB, 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 пакетов расширения системы MATLAB + Simulink.

Глава 3. Работа с файлами Simulink

- Интерфейс браузера библиотек
- Работа с браузером библиотек
- Интерфейс окна моделей Simulink
- Работа с окном моделей Simulink

Интерфейс браузера библиотек

Окно браузера библиотек

Библиотеки Simulink представляют собой обширный набор поименованных блоков, имеющих графическое изображение (пиктограмму) и содержащихся в отдельных файлах. Для их просмотра служит браузер библиотек.

Как уже отмечалось в главе 2, после запуска Simulink выводится окно браузера библиотек. На рис. 3.1 это окно показано отдельно. Окно имеет заголовок, меню, панель инструментов, поле информационных сообщений, окна с деревом библиотек и компонентами выделенного раздела библиотек и строку состояния (внизу окна). На рис. 3.1 выделена основная библиотека Simulink (в левом окне) и показаны ее разделы (в правом окне).

Если в левом окне выделен какой-то раздел библиотек, то щелчок правой кнопкой мыши выводит контекстное меню с одной командой, позволяющей вывести данный раздел в отдельном окне. В нашем случае это команда Open the "Simulink" Library (Открыть библиотеку Simulink). Окно основной библиотеки Simulink показано на рис. 3.2.

Как видно из рис. 3.1, перед именем каждого раздела библиотеки имеется прямоугольная кнопка со знаком +. Нажатие на нее ведет к раскрытию дерева раздела библиотеки. При этом в правом окне появляется набор компонентов этого раздела библиотеки (рис. 3.3).

ПРИМЕЧАНИЕ

Любопытно отметить, что данное окно (рис 3 2) относится к средствам, взятым из предшествующей версии Simulink. Поэтому в заголовке имеется ссылка на версию Simulink 3 Однако в самом окне представлены разделы версии Simulink 4.0. На практике целесообразно пользоваться браузером библиотек, но отдельные окна библиотек удобнее для общего обзора содержания различных разделов библиотек, особенно если они содержат много блоков

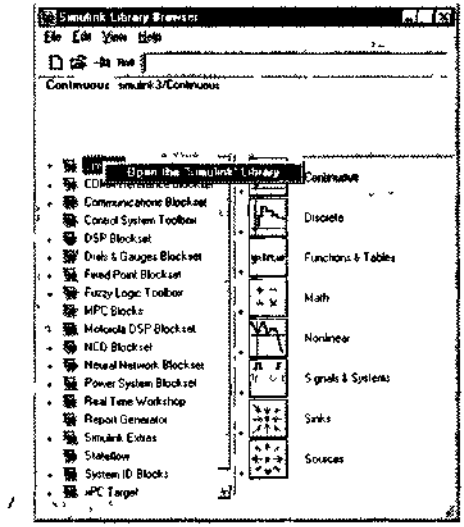


Рис. 3.1. Окно браузера библиотек Simulink

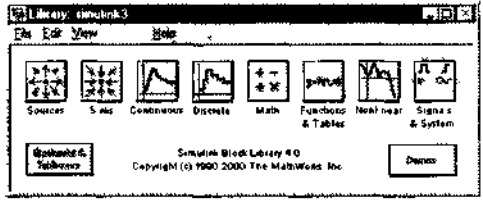


Рис. 3.2. Окно основной библиотеки Simulink

С помощью контекстного меню можно вызвать отдельное окно для раздела библиотеки (см. на рис. 3.3 в правой части окна браузера). Обратите внимание на то, что порядок расположения компонентов в этом окне несколько иной, чем в правом окне браузера библиотек.

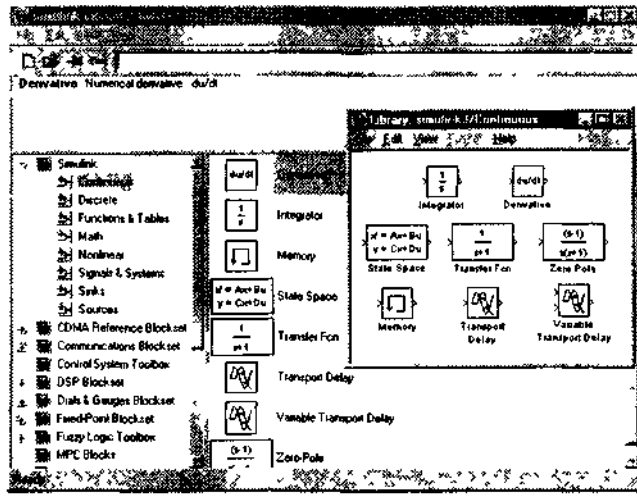


Рис. 3.3. Пример доступа к разделу библиотек Continuous

Из любого окна (браузера или отдельного окна раздела библиотеки) можно перетаскивать компоненты мышью в окно модели.

Заголовок и строка состояния

Окно браузера библиотек имеет такие стандартные для окон Windows элементы, как заголовок и строка состояния. Заголовок имеет кнопку, открывающую стандартное меню управления окном: восстановление окна в размерах, перемещение, изменение размера, свертывание и развертывание и закрытие.

Строка состояния служит для вывода коротких сообщений о состоянии системы. Эти сообщения контекстно-зависимые.

Меню окна браузера библиотек

Меню браузера библиотек содержит следующие пункты:

- File — работа с файлами библиотек;
- Edit — добавление компонентов и их поиск;
- View — управление показом элементов интерфейса;
- Help — вывод окна справки по браузеру библиотек.

На рис. 3.4 показано окно браузера библиотек с открытым пунктом меню File. Этот пункт меню имеет всего три команды: New — созда-

ние S-модели (команда Model) или библиотеки (команда Library), Open... — загрузка файлов S-модели и их поиск в файловой системе и Preferences — настройка Simulink.

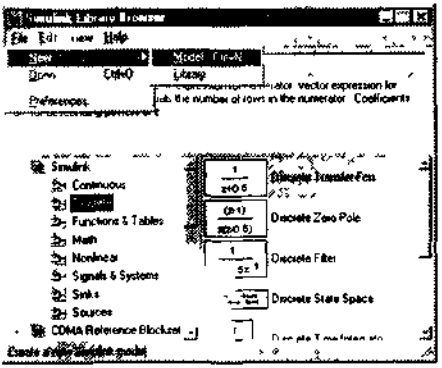


Рис. 3.4. Окно браузера библиотек — меню File

Создание пользователем своей Simulink-модели (S-модели) мы рассмотрим далее в главе 4. Команда File ▸ New ▸ Library позволяет создать библиотеку, в которую пользователь может поместить нужные ему компоненты. Действие команды File ▸ Open... рассмотрено в главе 2 на примере загрузки ряда демонстрационных примеров.

Настройка параметров Simulink

Команда File ▸ Preferences выводит окно установки ряда параметров системы MATLAB + Simulink. Здесь мы рассмотрим только те его возможности, которые относятся к пакету Simulink. Окно Preferences с открытой ветвью дерева параметров Simulink и панелью установки шрифтов Simulink Font Preferences показано на рис. 3.5. Установки в этом окне вполне очевидны — они относятся к выбору шрифтов для различных компонентов библиотек Simulink.

То же окно, но с панелью установок параметров моделирования Simulink Simulation Preferences показано на рис. 3.6. Эта панель имеет три вкладки:

- Solver — установка параметров решателя дифференциальных уравнений;
- Workspace — установка параметров рабочего пространства;
- Diagnostics — установка параметров диагностики.

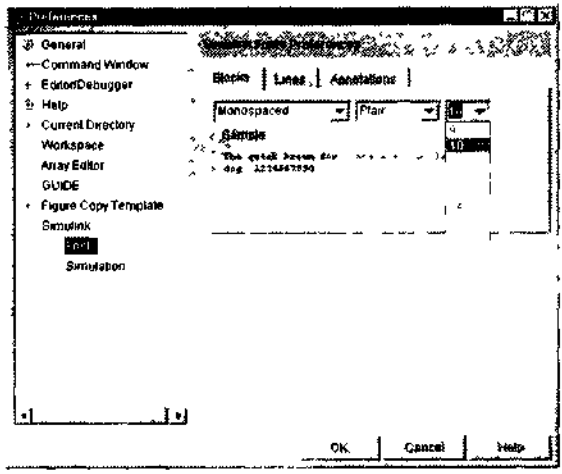


Рис. 3.5. Окно Preferences с установками шрифтов

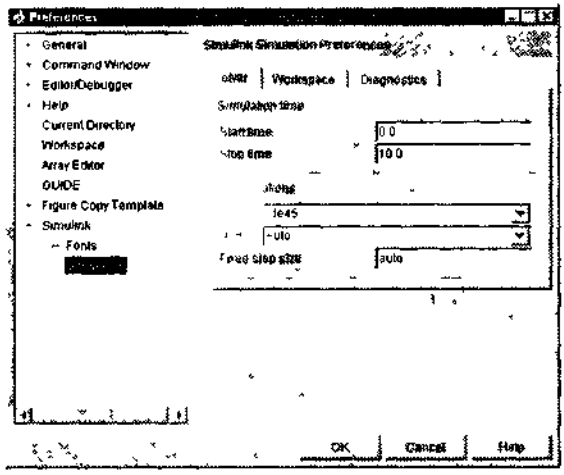


Рис. 3.6. Окно Preferences с установками шрифтов

Наиболее важные параметры устанавливаются на вкладке Solver. Прежде всего это выбор начального и конечного условного времени моделирования и выбор типа решателя дифференциальных уравнений. Назначение этих (и остальных) параметров мы уже обсуждали в главе 2. Вкладка Workspace имеет всего две опции, которые позволяют сохранить в рабочем пространстве время и состояние выхода модели. По умолчанию обе эти опции включены. Наконец,

вкладка Diagnostics также имеет всего две опции, устанавливающие тип диагностических ошибок.

Меню Edit браузера библиотек

В меню Edit браузера библиотек имеется три команды.

- Add to the current model — добавить выделенный блок в текущую модель;
- Find block... — найти блок с заданным именем;
- Find next block... — найти следующий блок с заданным именем.

Первая команда очевидна. Надо лишь отметить, что если текущего окна нет, то появится окно с предложением создать новое окно модели. Команда Find block... выводит окно с запросом имени блока (рис. 3.7).

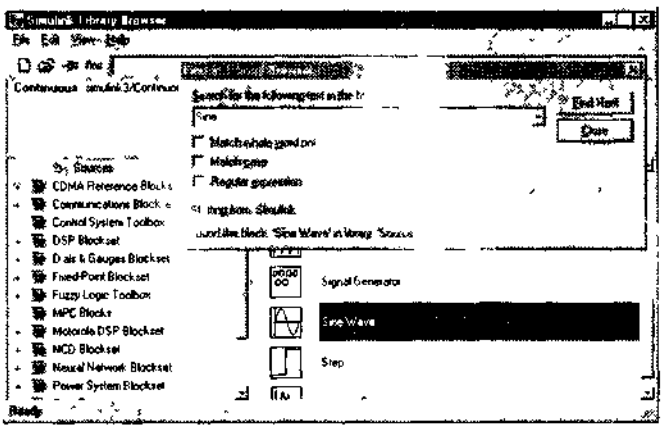


Рис. 3.7. Поиск блока по заданному имени

Команда Find next block... служит для поиска следующего блока, в который входит заданное слово. Эту же операцию выполняет и команда Find next в окне задания слова для поиска.

Меню View браузера библиотек

Меню View браузера библиотек содержит команды для управления видом интерфейса браузера:

- Toolbar — вывод/скрытие панели инструментов;
- Status bar — вывод/скрытие строки состояния;

- Description — вывод/скрытие окна сообщений;
- Stay on top — установка статуса окна браузера «поверх всех окон»;
- Collapse enteir Browser — закрытие текущего раздела библиотеки;
- Expand entire Browser — раскрытие текущего раздела библиотеки;
- Large icons — отображение пиктограмм блоков в увеличенном размере;
- Small icons — отображение пиктограмм блоков в уменьшенном размере;
- Show Parameters for selected block — вывод окна установки параметров отмеченного блока.

Справка по браузеру библиотек

Меню Help служит для доступа к справочной системе окна браузера. Содержит следующие команды:

- Help on the selected block — справка по выделенному блоку;
- Simulink help — вывод окна справочной системы Simulink;
- Tip of the day — полезные советы каждый день.

Использование этих средств справки вполне очевидно.

Панель инструментов окна браузера библиотек

Как видно из рис. 3.1, панель инструментов окна браузера библиотек имеет всего четыре кнопки, дублирующие некоторые команды меню:

- Create a new model — создание S-модели;
- Open a model — загрузка S-модели;
- Stay on top — установка статуса окна браузера «поверх всех окон»;
- Find — поиск блока по заданному имени.

Интерфейс окна моделей Simulink

Панель инструментов окна моделей

Команда New model в окне браузера библиотек открывает пустое окно модели пакета Simulink (рис. 3.8).

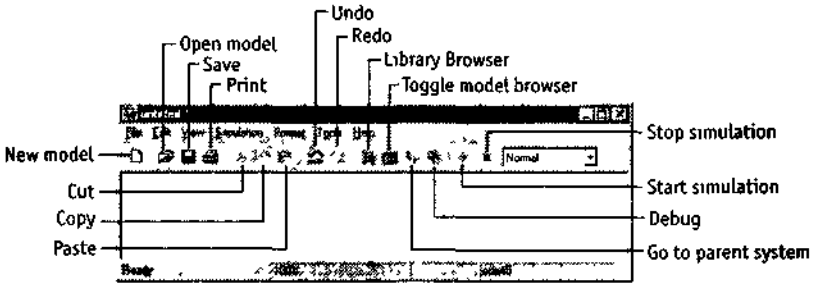


Рис. 3.8. Окно S-модели пакета Simulink

Как видно из рис. 3.8, панель инструментов окна модели содержит 15 кнопок. Эти кнопки имеют следующее назначение:

- New model — создание новой модели;
- Open model — загрузка модели;
- Save — сохранение текущей модели;
- Print — печать текущей модели;
- Cut — перенос выделенного объекта в буфер;
- Copy — копирование выделенного объекта в буфер;
- Paste — вставка объекта из буфера;
- Undo — отмена последней операции;
- Redo — восстановление последней отмененной операции;
- Library Browser — открытие браузера библиотек;
- Toggle model browser — открытие браузера моделей в левой части окна модели;
- Go to parent system — переход в основную (родительскую) систему;
- Debug — переход в режим отладки модели;
- Start simulation — запуск моделирования;
- Stop simulation — остановка моделирования.

Основное меню пакета Simulink

Несмотря на наличие панели инструментов, важная роль принадлежит обычному меню пакета Simulink. Оно дает более полный набор средств по управлению процессом моделирования. Меню (см. рис. 3.8) содержит следующие пункты:

- File — операции с файлами S-моделей;

- Edit — операции редактирования текущей модели;
- View — управление видом интерфейса;
- Simulation — операции запуска моделирования и его настройки;
- Format — операции форматирования текущей модели;
- Tools — доступ к инструментальным средствам;
- Help — доступ к средствам справочной системы.

Меню File

Меню File содержит ряд команд для работы с файлами (рис. 3.9).

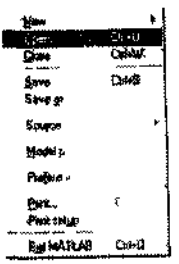


Рис. 3.9. Команды меню File

- New — создание модели (Model) или библиотеки (Library);
- Open... — загрузка файла модели;
- Close — закрытие окна текущей модели;
- Save — сохранение текущей модели;
- Save As ... — сохранение текущей модели под заданным именем;
- Source control... — управление источниками сигналов;
- Model properties... — вывод окна свойств текущей модели;
- Preferences... — вывод окна свойств Simulink;
- Print... — печать текущей модели;
- Print setup — вывод окна установок печати;
- Exit MATLAB — завершение работы.

Окна управления источниками сигналов

Команда Source Control... выводит подменю с командами Check in... (Проверка входа), Check out (Проверка выхода), Undo Check out (Отмена проверки выхода) и Preferences (Вывод окна настроек). Первые две

команды выводят окна Check in и Check out, которые видны на рис. 3.10. Как правило, в них задаются опции по умолчанию.

Окна Check In и Check out позволяют ввести расширенное текстовое описание источника. Команда Preferences выводит окно настроек пакета Simulink (окно Preferences) с открытой панелью General ▸ Source control, что позволяет выбрать схему управления источниками.

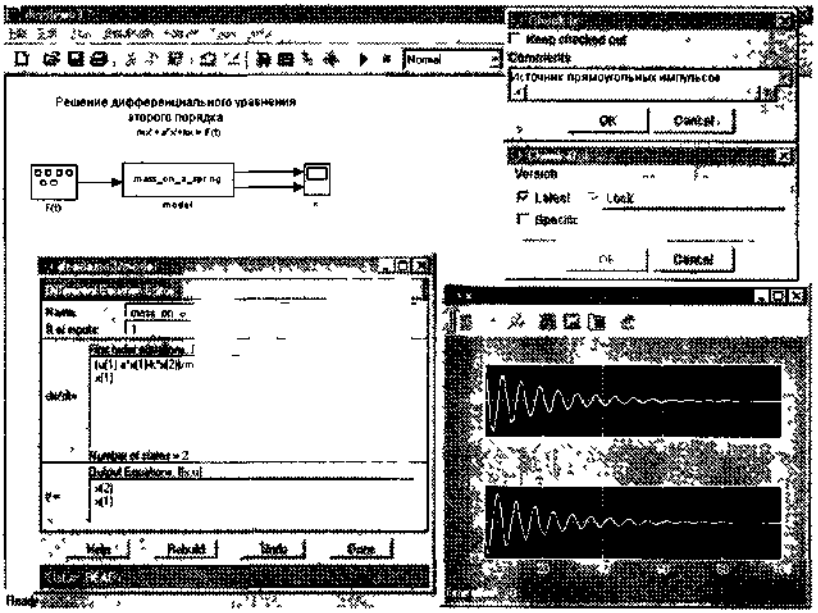


Рис. 3.10. Выбор схемы управления источниками сигналов

Обратите внимание на то, что рис. 3.10 дает еще один пример решения дифференциального уравнения второго порядка, представленного эквивалентной системой из двух дифференциальных уравнений. Решение этого уравнения описывает затухающий процесс колебаний массивного маятника.

Вывод окна свойств текущей модели

Команда Model Preferences выводит окно свойств текущей модели (рис. 3.11). Это окно имеет три вкладки: Model Properties, Options и Histoty. Вкладка Model Properties отображает данные о номере версии

модели, ее разработчике и дате создания. В ней можно задать описание модели (обратите внимание, что это возможно на русском языке).

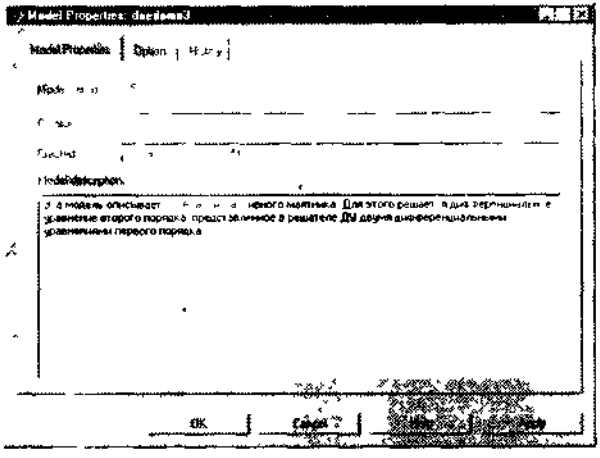


Рис. 3.11. Вкладка Model Properties окна свойств модели

Вкладка Options позволяет задать несколько параметров, которые, как правило, используются по умолчанию. Назначение параметров можно понять из рис. 3.12. Задаются тип менеджера конфигурации (выбор из раскрывающегося списка) и строковые записи форматов версии модели, ее модификации и даты модификации.

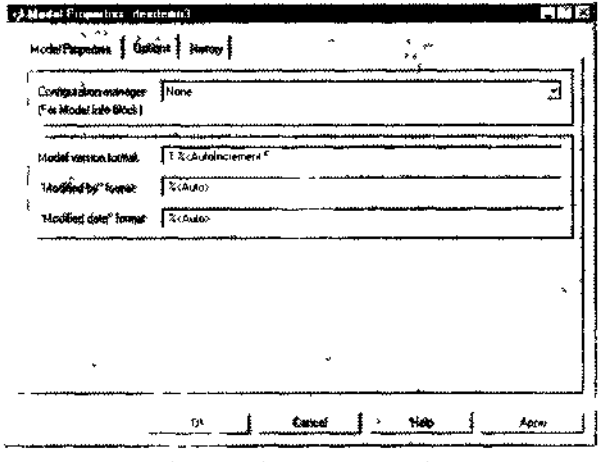


Рис. 3.12. Вкладка Options окна свойств модели

Вкладка History (рис. 3.13) несет данные о модификации системы — если, разумеется, таковые есть. Кнопка Edit открывает доступ к окну сохранения модификаций модели (рис. 3.13).

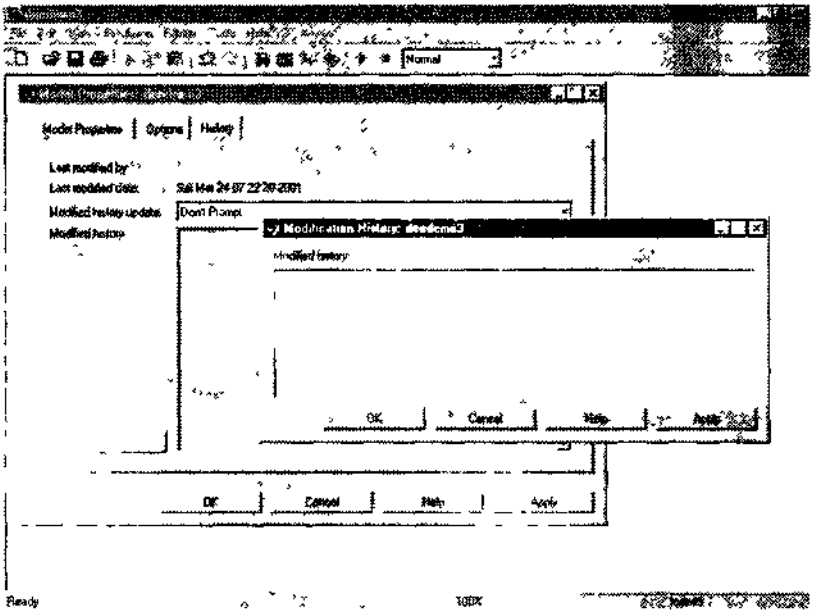


Рис. 3.13. Окно свойств модели с открытой вкладкой History

К возможности изменения свойств модели прибегают обычно только опытные пользователи — обычно бывает достаточно установок свойств по умолчанию.

Печать текущей модели

Команда File ▸ Print... выводит окно печати модели. Это окно (рис 3.14) содержит все необходимые настройки для печати текущей модели. Прежде всего это выбор типа принтера для печати, область печати и выбор опций печати.

Выбор принтера возможен, если в системе Windows 95/98/NT/2000 установлен ряд драйверов принтера. Опции печати задают объем печати, число копий, схему печати (определяющую глубину распечатки данных о блоках), печать фрейма модели и др. Особое внимание стоит обратить на выбор схемы печати и на возможность печати модели с разным уровнем вложения субмоделей (подсистем).

Кнопка Properties в правом верхнем углу окна печати выводит окно свойств выбранного принтера. На рис. 3.13 представлено окно свойств принтера для цветного струйного принтера EPSON Stylus COLOR 600. Это стандартное окно Windows, и его вид зависит от драйвера заданного принтера. Для указанного принтера окно содержит три вкладки для установки текущих параметров печати, параметров бумаги и запуска утилит профилактики принтера.

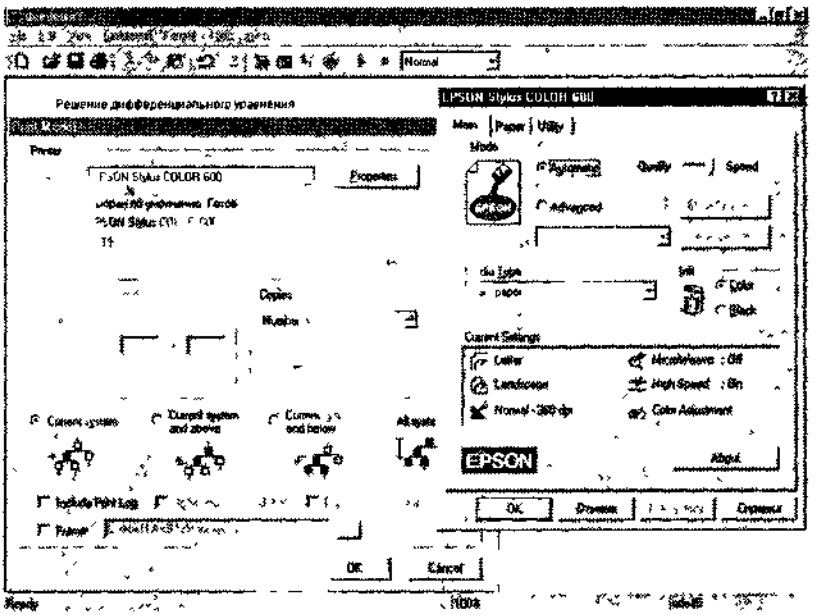


Рис. 3.14. Окно печати текущей модели

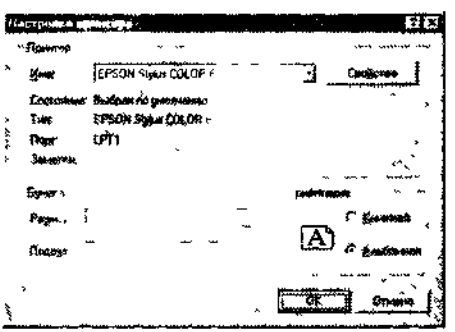


Рис. 3.15. Окно настройки принтера

Настройка принтера

Команда Print Setup... в меню File окна Simulink открывает окно настройки принтера (рис. 3.15). Это окно операционной системы Windows, поэтому если она русифицирована, то не стоит удивляться появлению в окне русскоязычных надписей.

Параметры этого окна очевидны. Кнопка Свойства открывает окно свойств выбранного принтера, которое уже приводилось (см. рис. 3.13).

Глава 4. Подготовка и запуск модели

- Создание модели
- Моделирование ограничителя
- Основные приемы подготовки и редактирования модели
- Операции форматирования модели

Создание модели

Постановка задачи и начало создания модели

Решение любой проблемы в системе Simulink должно начинаться с постановки задачи. Чем глубже продумана постановка задачи, тем больше вероятность успешного ее решения. В ходе постановки задачи нужно оценить, насколько суть задачи отвечает возможностям пакета Simulink и какие компоненты последнего могут использоваться для построения модели.

Основные команды редактирования модели сосредоточены в меню Edit. В качестве примера применения этих команд рассмотрим построение простой модели, а точнее, сразу трех простых моделей в пределах одного окна. Это, кстати, будет весьма поучительный эксперимент, показывающий, что можно одновременно моделировать несколько систем.

Итак, сначала откроем пустое окно для новой модели (кнопка Create a new model в панели инструментов браузера библиотек Simulink).

Ввод текстовой надписи

Введем заголовок нашей будущей модели — «Simple model» (Простая модель). Для этого достаточно установить курсор мыши в нужное место окна и дважды щелкнуть левой кнопкой мыши. Появится

прямоугольная рамка, внутри которой находится мигающий маркер ввода в виде вертикальной палочки.

Теперь можно ввести нужную надпись по правилам, действующим для строчного редактора. Пока будем считать, что параметры надписи по умолчанию нас вполне устраивают.

Размещение блоков в окне модели

Из раздела библиотеки Sources перетащим мышью три источника сигнала: синусоидального, прямоугольного (дискретного) и пилообразного. Затем из раздела Sinks перетащим в окно модели блок осциллографа. В результате получим модель в виде, представленном на рис. 4.1.

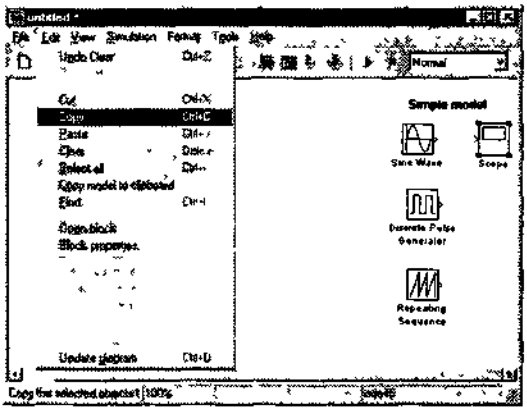


Рис. 4.1. Начало подготовки модели

Выделение блока модели

На рис. 4.1 показано также меню редактирования Edit в открытом виде — при выделении блока в этом меню становятся доступны команды редактирования свойств блока. Для выделения блока достаточно навести на него маркер мыши и нажать левую кнопку. В рамке блока по углам появятся маленькие темные прямоугольники, которые и являются признаком того, что блок выделен. На рис. 4.1 выделен блок осциллографа Scope.

Если захватить курсором мыши уголок выделенного блока, то можно заметить, что курсор мыши превратится в перекрестие тонких диагональных двухсторонних стрелок. Это означает, что можно про-

порционально увеличивать или уменьшать блок в диагональных направлениях.

Меню редактирования Edit

Кратко рассмотрим основные команды меню Edit (рис. 4.1). Это меню содержит ряд типовых команд, которые разбиты на 6 групп. В первой группе есть две команды: Undo (отмена последней операции) и Redo (восстановление последней отмененной операции). Эти команды являются контекстно-зависимыми.

Следующая группа команд связана с операциями с буфером обмена Windows:

- Cut — перенос выделенных объектов в буфер,
- Copy — копирование выделенных объектов в буфер;
- Paste — вставка объектов из буфера в заданное курсором мыши место;
- Clear — уничтожение выделенных объектов;
- Select All — выделение всех объектов модели;
- Copy model to clipboard — копирование всей модели в буфер;
- Find — поиск в модели заданного объекта.

Остальные команды подменю Edit носят специальный характер и будут рассмотрены в дальнейшем.

Применение буфера обмена

Вернемся к нашей модели и покажем некоторые приемы работы с буфером обмена. На рис 4.1 выделен блок осциллографа Scope. После выполнения команды Copy копия выделенного блока Scope поступает в буфер обмена и хранится в нем. При выполнении команды Cut помещенный в буфер блок исчезает из окна модели.

Теперь для вставки копии блока Scope достаточно поместить в нужное место курсор мыши и выполнить команду Paste. Блок Scope1 появится в указанном месте (рис. 4.2). Аналогично добавляется еще один блок — Scope2 — к третьему источнику сигнала (рис. 4.3).

ВНИМАНИЕ

Обратите внимание на то, что на самом деле команда Paste в нашем случае не дает строгого переноса блоков осциллографа. Каждый новый блок автоматически получает новое наименование. Так, если исходный блок назывался Scope, то следующий становится Scope1, затем Scope2 и т. д.

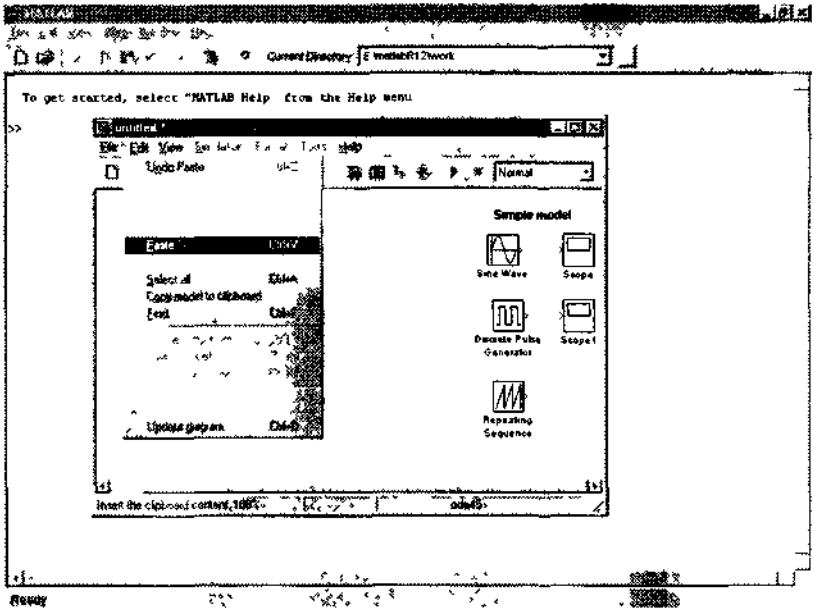


Рис. 4.2. Размножение блока Scope с помощью буфера обмена

Теперь можно приступить к соединению выходов источников со входами осциллографов. Для этого достаточно указать курсором мыши на начало соединения (выход источника) и затем при нажатой левой кнопке мыши протянуть соединение в его конец (вход осциллографа). В итоге получим модель, показанную на рис. 4.3.

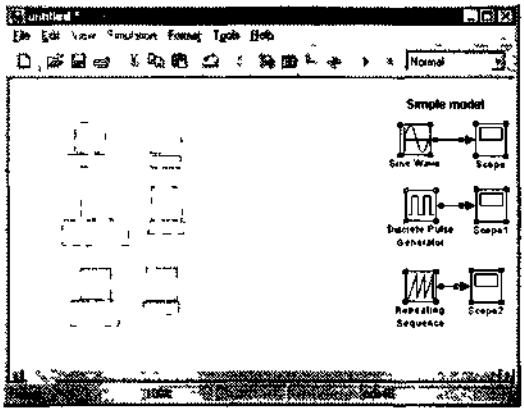


Рис. 4.3. Готовая модель

Выделение ряда блоков и их перенос

Блоки нашей модели размещаются в правой части окна модели. Допустим, мы задумали перенести их все вместе в левую часть окна. Для этого надо выделить все блоки. Это можно сделать двумя способами.

Первый способ — воспользоваться командой Select all. Второй способ основан на работе с мышью. Нужно установить курсор мыши рядом с выделяемыми блоками и нажать ее левую кнопку. Теперь при перемещении мыши появится расширяющаяся прямоугольная рамка из тонких пунктирных линий Как только в ней окажется какой-либо блок, он будет выделен. Охватив рамкой все блоки, можно выделить их все.

Выделенный набор блоков можно перетаскивать мышью как единый объект. Это показано на рис. 4.3, где справа виден набор исходных блоков, а слева — перетаскиваемый блок. Отпустив левую кнопку мыши, можно увидеть блоки на новом месте.

Запуск нескольких моделей одновременно

Теперь все готово для нашего первого серьезного эксперимента — одновременного запуска нескольких моделей. Чтобы получить приведенные далее результаты, необходимо установить параметры Start time=0 и StopTime=20 в окне установки параметров моделирования (напоминаем, что оно вызывается командой Simulation ▸ Simulation parameters...). После этого, запустив моделирование нажатием кнопки Start Simulation или командой меню Simulation ▸ Start, можно увидеть результат, показанный на рис. 4.4.

На самом деле вначале картина не меняется и лишь спустя секунду-другую слышится звук колокола — он свидетельствует об окончании процесса моделирования. Чтобы получить осциллограммы от каждого из осциллографов, надо активизировать их, сделав на каждом из них двойной щелчок мышью При этом появятся их осциллограммы в произвольных местах экрана Полученные таким образом осциллограммы можно перетащить мышью в удобные для обзора положения. Их можно также растянуть или сжать в любом направлении с помощью мыши. В результате мы получим вид экрана, представленный на рис. 4.4.

Итак, мы видим, что все три модели работают и осциллограммы представляют временные зависимости сигналов, которые вырабатыва-

ют источники, — синусоиду, прямоугольные импульсы и треугольные импульсы.

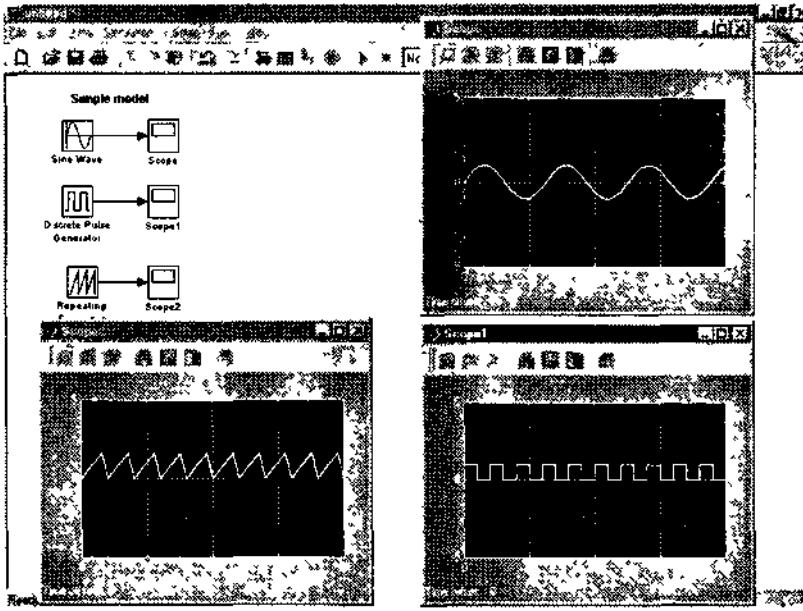


Рис. 4.4. Готовая модель и осциллограммы ее работы

Моделирование ограничителя

Постановка задачи

В качестве следующего примера рассмотрим тривиальную задачу моделирования работы идеального ограничителя, на вход которого подается синусоидальное напряжение с амплитудой 5 В и частотой 1 рад/с. Допустим, что пороги ограничения составят +0.5 и –0.5 В. Заметим, что такие параметры источник синусоидального сигнала имеет по умолчанию.

В данном случае очевидно, что основными блоками будут генератор синусоидальных сигналов и блок нелинейности, моделирующий передаточную характеристику ограничителя. Кроме того, к этим блокам надо добавить регистрирующий блок — осциллограф. Так как функциональная схема моделируемого устройства в данном случае вполне очевидна, то мы можем перейти к ее реализации.

Создание модели ограничителя

Для создания модели данного устройства проделаем следующие действия:

1. Откроем окно новой модели Simulink, нажав кнопку Create a new model.
2. Расположим это окно рядом с окном браузера библиотек.
3. Из раздела библиотеки Sources перенесем в окно модели источник синусоидального сигнала Sine Wave.
4. Из раздела библиотеки Nonlinear перенесем в окно модели нелинейный блок — ограничитель Saturation.
5. Из раздела библиотеки Sinks перенесем в окно модели блок осциллографа Scope.
6. Выполним соединение между блоками (как это сделать, описано в предыдущем примере).
7. Проверим установку времени моделирования: Start time=0 и Stop time=20.
8. Щелкнув дважды по блоку Sine Wave, в появившемся окне параметров источника синусоидального сигнала установим амплитуду равной 5.
9. Запустим модель на исполнение, нажав кнопку Start Simulation в панели инструментов окна модели.

Результат представлен на рис. 4.5.

Настройка масштаба осциллограмм

Нетрудно заметить, что масштаб отображения осциллограммы у осциллографа на рис. 4.5 оказался не совсем удачным — изображение осциллограммы мало по высоте, поскольку при порогах в 0.5 масштаб в 5 условных единиц уровня получается слишком крупным. Заметим, что мы не указываем размерность осциллограммы по вертикали. В зависимости от условий задачи это могут быть метры (задача на движение), вольты (электронный ограничитель) и т. д.

Для смены масштаба достаточно щелкнуть правой кнопкой мыши в окне осциллограммы. В появившемся контекстном меню (рис. 4.6) нужно выбрать команду Axes Properties... , которая служит для задания масштаба осциллограммы.

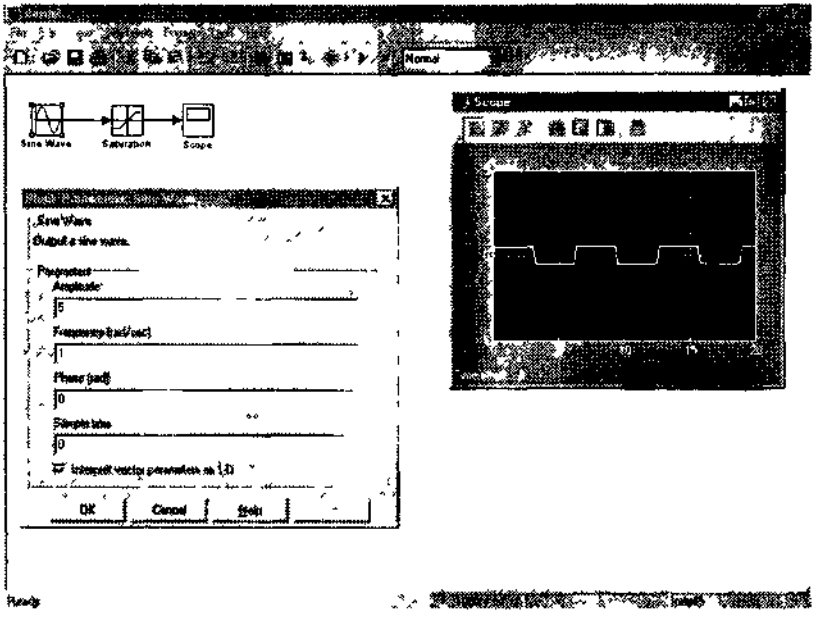


Рис. 4.5. Модель ограничителя и результат ее работы

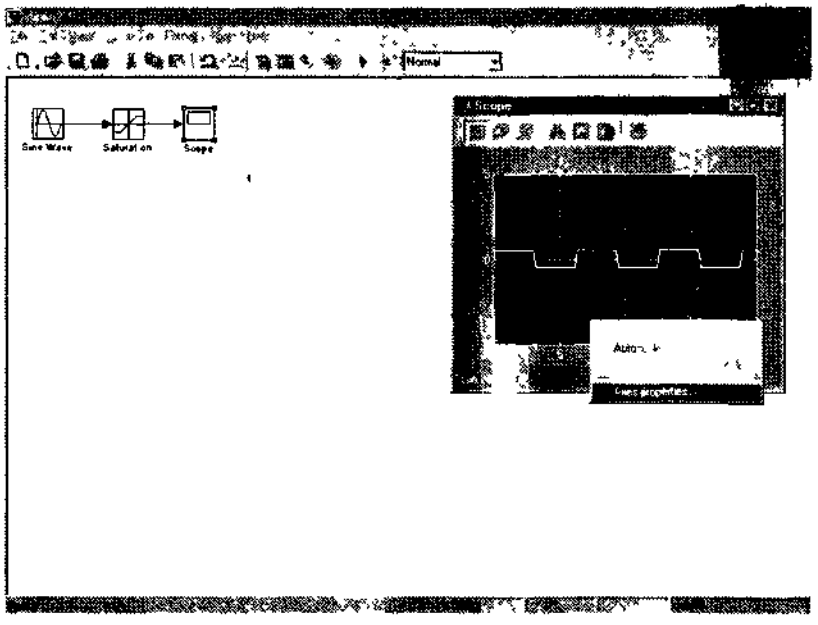


Рис. 4.6. Контекстное меню окна осциллограммы

В открывшемся окне свойств осей надо заменить значения Y-min=-5 и Y-max=5, например на Y-min=-0.8 и Y-max=0.8. После этого, нажав кнопку Apply, можно увидеть осциллограмму с измененным масштабом (рис. 4.7).

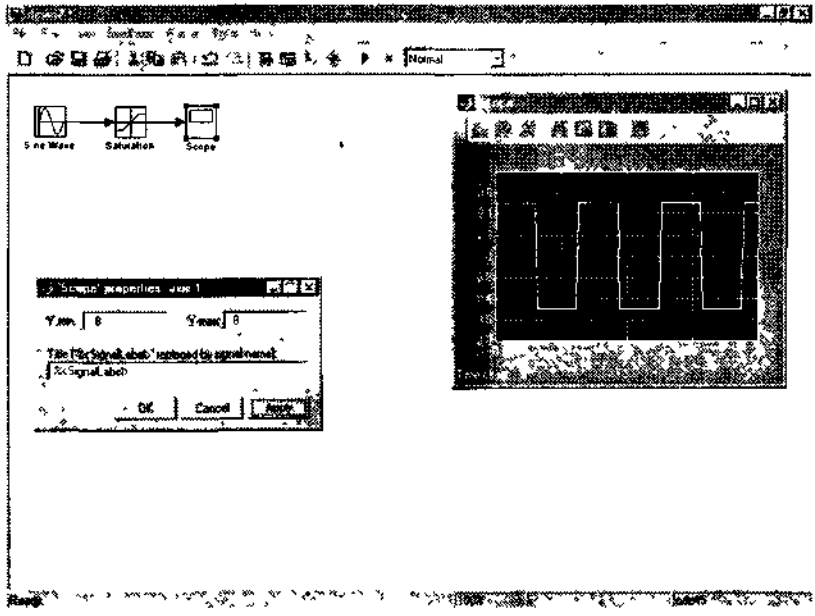


Рис. 4.7. Осциллограмма в новом масштабе

Нетрудно заметить, что теперь осциллограмма стала гораздо более представительной. Если вид осциллограммы вас устраивает, то достаточно зафиксировать сделанные изменения масштаба, нажав кнопку OK в окне свойств осей. Можно и отказаться от сделанных изменений, нажав кнопку Cancel.

На рис. 4.8 в контекстном меню осциллограммы видна еще команда — Autoscale (Автомасштабирование). Эта же команда реализуется кнопкой Autoscale в панели инструментов окна осциллограммы. Эта команда устанавливает масштаб, при котором окно осциллограммы используется полностью. В данном случае это означает, что осциллограмма будет иметь максимально возможный размер, как показано на рис. 4.8.

Итак, как и следовало ожидать, в результате моделирования получена синусоида с обрезанными на уровне 0.5 сверху и снизу верши-

нами. При этом результат получен мгновенно — см. данные в строке состояния окна модели (время моделирования — 0). Столь быстрое получение явно верного результата достигается далеко не всегда. Чем сложнее модель, тем больше усилий и времени придется затратить на то, чтобы добиться ее правильной работы.

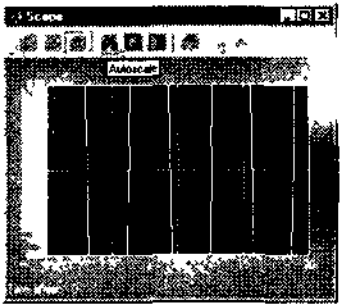


Рис. 4.8. Осциллограмма после выполнения команды Autoscale

Сохранение модели

Можно сохранить созданную модель для последующего применения, показа или модернизации. Для этого используется команда Save или Save As меню File окна редактора моделей. Модель записывается в виде файла с расширением .mdl.

Модернизация и расширение модели

Теперь, после создания простой модели, можно попытаться модернизировать ее и дополнить, изменить параметры модели и т. д. На рис. 4.9, например, показана модель, в которой к источнику синусоидального сигнала подключены уже четыре нелинейных устройства. Это ограничитель пиков, нелинейность типа «мертвая зона», квантующее устройство и релейный пороговый элемент (в электронике известный под именем триггера Шмитта). К каждому из них подключен свой осциллограф.

Как видно из рис. 4.9, теперь можно наблюдать сигналы с выходов каждого из четырех нелинейных устройств, причем с помощью отдельных осциллографов. Позже мы покажем, что возможно наблюдение и нескольких сигналов одним осциллографом, если перед ним установить мультиплексор сигналов.

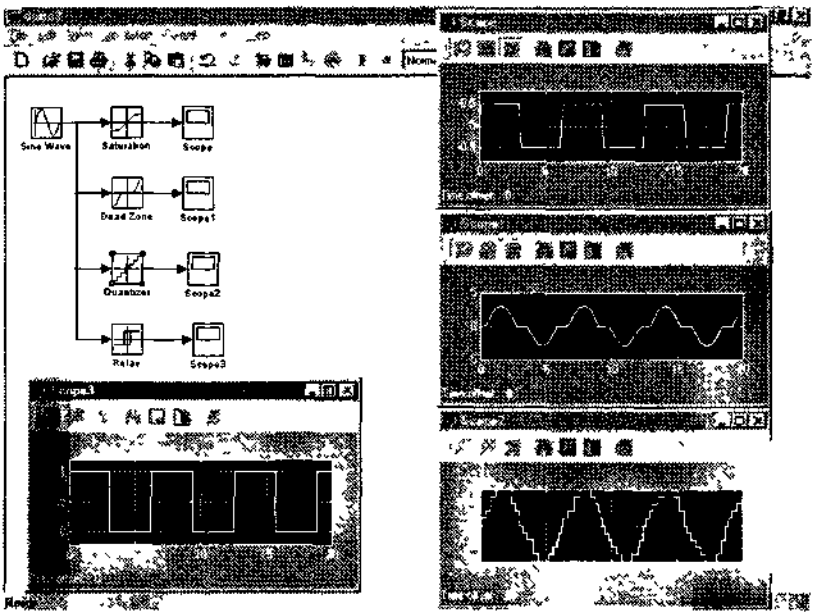


Рис. 4.9. Модель из трех нелинейных устройств

Основные приемы подготовки и редактирования модели

Рассмотрев простейшие примеры моделирования, приступим к систематизированному описанию приемов подготовки и редактирования моделей.

Добавление надписей и текстовых комментариев

Модели, не содержащие текстовых комментариев, не наглядны. Зачастую без комментариев даже трудно понять, что, собственно говоря, моделируется. Поэтому подготовка текстовых комментариев — важный момент в культуре моделирования, кстати, как и в программировании.

Как уже отмечалось, создать надпись в окне модели очень просто. Достаточно указать мышью место надписи и дважды щелкнуть левой кнопкой мыши, и появится блок надписи с курсором ввода (рис. 4.10).

Так же просто можно изменить подписи к блоками моделей. Для этого нужно установить мышь в область надписи и щелкнуть левой кнопкой мыши — в подписи появится курсор ввода и ее можно будет редактировать. На рис. 4.11 показано изменение надписи в блоке осциллографа.

Это текстовая надпись

Рис. 4.10. Ввод текстового комментария

Рис. 4.11. Изменение подписи к блоку

Чтобы убрать надпись, нужно выделить ее (кстати, как и любой другой объект) и выполнить команду Edit ▸ Clear.

ВНИМАНИЕ

Из приведенных примеров может показаться, что можно запросто переделать все англоязычные надписи на русскоязычные. Однако это не совсем так. Некоторые операции (например, выравнивание подписей и даже заполнение текстового блока) при использовании символов кириллицы работают не вполне корректно. Может наблюдаться и самопроизвольная смена шрифта при завершении создания надписи. Удивляться этому не стоит — MATLAB не относится к продуктам, локализованным в России. Поэтому без веских оснований не стоит пользоваться символами кириллицы.

Выделение, удаление и восстановление объектов

Выделение объектов удобнее всего осуществляется мышью. Достаточно установить курсор мыши на нужном объекте и щелкнуть левой кнопкой мыши. Объект будет выделен. Мышью также можно выделить несколько объектов. Для этого надо установить курсор мыши вблизи них, нажать левую кнопку мыши и, удерживая ее, начать перемещать мышь. Появится динамическая пунктирная рамка. Все попавшие в нее объекты становятся выделенными. Выделить все объекты также можно, используя команду Edit ▸ Select All.

Для стирания выделенного объекта можно вызвать команду Clear из меню Edit или из контекстного меню (рис. 4.12). Контекстное меню очень удобно тем, что для любого объекта оно выводит перечень команд, которые доступны в данном состоянии.

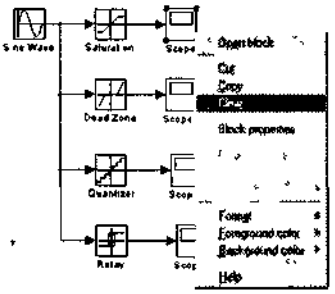


Рис. 4.12. Контекстное меню

Для восстановления объекта в окне модели следует щелкнуть левой кнопкой мыши в предполагаемом месте расположения объекта. После этого выполнение команды Paste из меню File окна Simulink или из контекстного меню помещает хранящийся в буфере объект (блок) в заданное место

Следует учесть, что команда Clear стирает блок безвозвратно, то есть без помещения его в буфер обмена. Однако эту операцию можно отменить командой меню File ▸ Undo окна Simulink

Вставка блоков и их соединение

Вставку блоков с помощью браузера библиотек Simulink мы уже обсудили достаточно подробно в примерах. Отметили также, что для переноса блоков, их копирования и размножения целесообразно использовать буфер обмена. Весьма плодотворным является подход, когда пользователь для создания своей модели использует ранее составленную модель — например, из отлаженных демонстрационных примеров, которых в пакете Simulink великое множество

Для подключения новых блоков нужны новые соединения. Они также легко выполняются с помощью мыши Вообще говоря, приемы ввода новых блоков и их соединений выполняются очень просто и естественно. При этом приемы редактирования очень напоминают работу с популярными графическими редакторами, которую легко осваивают даже дети.

Тем не менее отметить важнейшие приемы осуществления соединений полезно. Блоки моделей обычно имеют входы и выходы. Как правило, выход какого-либо блока подключается ко входу следую-

щего блока и т д Для этого курсор мыши устанавливается на выходе блока, от которого должно исходить соединение. При этом курсор превращается в большой крестик из тонких линий. Держа нажатой левую кнопку мыши, надо плавно переместить курсор ко входу следующего блока (рис. 4 13), где курсор мыши приобретет вид крестика из тонких сдвоенных линий.

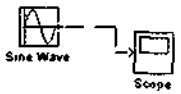


Рис. 4.13. Начало соединения блока источника с блоком осциллографа

Добившись протяжки линии ко входу следующего блока, надо отпустить левую кнопку мыши. Соединение будет завершено, и в конце его появится жирная стрелка. Щелчком мыши можно выделить соединение, признаком чего будут черные прямоугольники, расположенные в узловых точках соединительной линии (рис. 4.14).

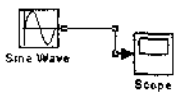


Рис. 4.14. Выделенное соединение

Иногда бывает нужно сделать петлю соединительной линии в ту или иную сторону. Для этого нужно захватить нужную часть линии и отвести ее в нужную сторону, перемещая мышь с нажатой левой кнопкой Рисунок 4 15 поясняет этот процесс

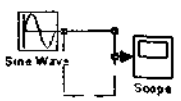


Рис. 4.15. Начало создания петли линии соединения

Создание петли линии заканчивается отпусканием левой кнопки мыши Полученная таким образом линия показана на рис. 4.16.

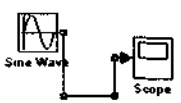


Рис. 4.16. Готовая петля соединения

Особо стоит отметить возможность задания наклонных линий соединений при нажатой клавише Shift. Пример такого соединения дан на рис. 4.17.

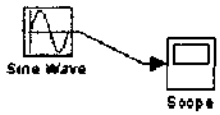


Рис. 4.17. Пример соединения с наклонной линией

Создание отвода линии

Часто возникает необходимость сделать отвод от уже созданной линии. Пример создания такого отвода иллюстрирует рис. 4.18. Заметим, что при нажатой клавише Shift отвод строится наклонными линиями.

В примере, показанном на рис. 4.18, использована модель интегратора, подключенного к выходу источника прямоугольных импульсов. Чтобы можно было наблюдать осциллограммы как на выходе источника, так и на выходе интегратора, в схему включен блок мультиплексора сигналов Mux с двумя входами. Чтобы подключить нижний вход к уже задействованному выходу источника, нам и понадобилось создать отвод линии.

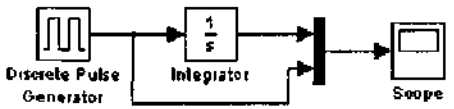


Рис. 4.18. Пример модели с отводом линии

Теперь можно запустить эту модель и посмотреть, какие сигналы действуют на выходах источника и интегратора. Результат запуска представлен на рис. 4.19.

Нетрудно убедиться в том, что сигнал на выходе интегратора представляет собой ступенчато нарастающую линию. Когда на выходе генератора имеется высокий (условно) уровень напряжения, на выходе интегратора сигнал линейно нарастает. Когда уровень на генераторе равен 0, сигнал на выходе интегратора остается неизменным. Такой характер процесса, разумеется, хорошо знаком специалистам. Но начинающим изучать электронные (и не только) системы этот пример дает наглядную иллюстрацию работы интегратора.

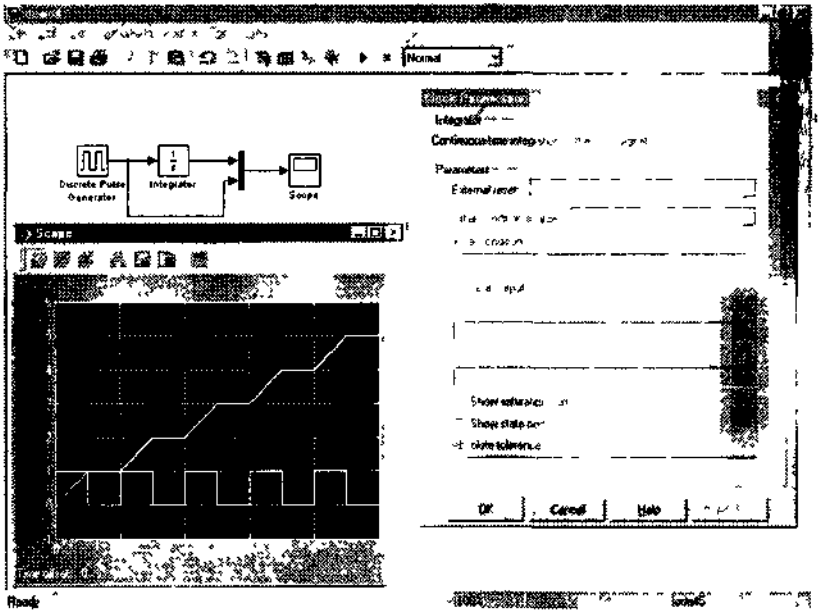


Рис. 4.19. Результат моделирования

Удаление соединений

Для удаления соединительной линии достаточно выделить ее и выполнить команду Clear или Cut.

Изменение размеров блоков

Simulink имеет расширенные возможности редактирования блок-схем. Так, блоки в окне редактирования можно не только перемещать с помощью мыши, но и изменять в размерах. Для этого блок выделяется, после чего курсор мыши надо установить на кружки по углам блока. Как только курсор мыши превратится в двунаправленную диагональную стрелку, можно будет при нажатой левой кнопке растягивать блоки по диагонали, увеличивая или уменьшая их размеры, — рис. 4.20.

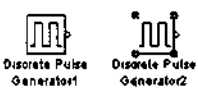


Рис. 4.20. Растяжение блока

Увеличенный в размерах блок показан на рис. 4.21. Обратите внимание на то, что растягивается только графическое изображение (пиктограмма) блока, а размеры его названия в виде текстовой надписи не изменяются.

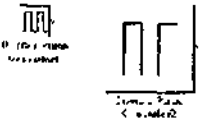


Рис. 4.21. Пример растяжения блока

Перемещение блоков и вставка блоков в соединение

Блок, участвующий в соединении, можно перемещать в окне модели, выделив его и перетаскивая, как обычно, мышью. При этом соединение не прерывается, а просто сокращается или увеличивается в длине. В длинное соединение можно вставить новый блок, не разрушая его и не выполняя сложных манипуляций. Рисунок 4.22 показывает вставку блока дифференцирующего устройства между источником синусондального сигнала и осциллографом.

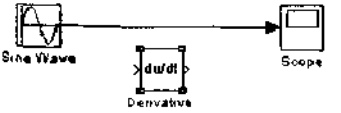


Рис. 4.22. Исходное соединение и блок для вставки

Результат вставки дифференцирующего устройства в соединение между источником и осциллографом показан на рис. 4.23.

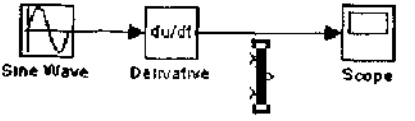


Рис. 4.23. Пример вставки блока в соединение

Однако подобная простая вставка возможна для блоков, имеющих один вход и один выход, которые включаются в соединение. Попытка вставить таким образом мультиплексор будет безуспешной, поскольку он имеет два входа и не стыкуется с разрываемым соединением.

Чтобы вставить мультиплексор, следует удалить соединение между дифференцирующим устройством и входом осциллографа. Для этого соединение выделяется и выполняется команда Edit ▸ Clear. После этого мультиплексор перемещается в нужное место и соединения создаются заново.

Моделирование дифференцирующего устройства

Итак, мы фактически создали модель дифференцирующего устройства и можем посмотреть, что происходит при дифференцировании синусоидального сигнала. Результат запуска созданной модели представлен на рис 4.24.

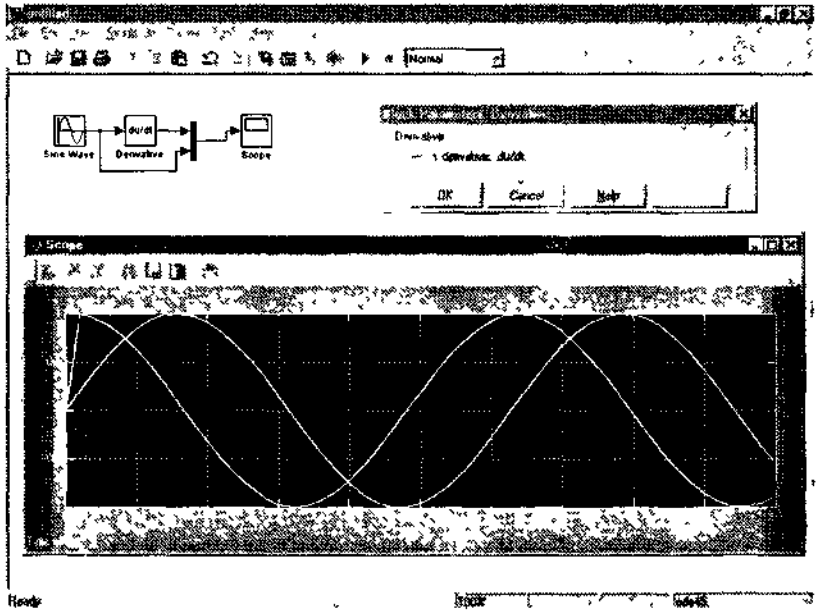


Рис. 4.24. Моделирование дифференцирующего устройства

Внимательно присмотревшись к осциллограммам, мы видим, что при входном синусоидальном сигнале выходной сигнал является косинусоидой. Это вполне отвечает математическим соотношениям для данного случая (как известно, производная $\sin(x)$ есть $\cos(x)$).

Однако в самом начале процесса дифференцирования хорошо виден изъян работы модели — при t = 0 производная равна не 1, а 0.

Это связано с тем, что процесс начинается при нулевых начальных условиях. Но довольно быстро ситуация исправляется, и в дальнейшем выходной сигнал становится косинусоидальным. Таким образом, дифференцирующее устройство можно использовать для точного сдвига на 90° гармонического сигнала любой частоты.

Обратите внимание, что в отличие от блока интегратора (рис. 4.19) блок цифрового дифференциатора не имеет настраиваемых параметров. Его окно параметров, показанное на рис. 4.24, является чисто информационным.

Команды Undo и Redo в окне модели

Большую помощь в редактировании оказывает команда Undo — отмена последней операции. Она поддерживает свыше ста различных операций, включая операции добавления и стирания линий. Эту команду можно вызвать с помощью кнопки в панели инструментов окна модели или из меню Edit. Для восстановления отмененной операции служит команда Redo.

Операции форматирования модели

Меню форматирования Format

В меню Format (и в контекстном меню) находится ряд команд форматирования блоков. Их можно разделить на несколько характерных групп.

- Управление отображением надписей и видом блоков:
  - Font — установка шрифта для текстовых надписей;
  - Text alignment — выравнивание текста в текстовом блоке;
  - Flip name — помещение подписи блока сверху или снизу блока;
  - Show/Hide name — отображение или скрытие подписи выделенного блока;
  - Flip block — отражение блока относительно вертикальной оси;
  - Rotate block — вращение блока на 90°;
  - Show drop shadow — показ тени от блока.
- Показ меток портов:
  - Show port labels — показ меток портов.

- Установка цветов:
  - Foreground color — установка цвета линий выделенных блоков;
  - Background color — установка цвета фона для выделенных блоков;
  - Screen color — установка цвета фона для всего окна модели.
- Прочие установки:
  - Library link display — отображение связей с библиотеками;
  - Simply time colors — установка цвета блока индикации времени;
  - Wide nonscalar lines — увеличение/уменьшение ширины нескалярных линий;
  - Signal dimensions — отображение размерности сигналов;
  - Port data types — вывод данных о типе портов;
  - Execution order — вывод порядкового номера блока в последовательности исполнения.

Команда Format ▸ Font выводит окно с установками шрифта для текстовых надписей (рис 4.25).

Рисунок 4.26 показывает операции, связанные с изменением положения блоков. Это команды Flip block и Rotate block.

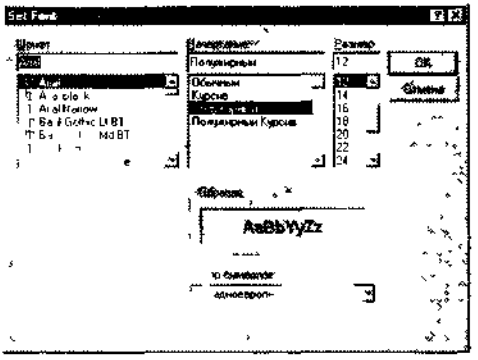


Рис. 4.25. Окно выбора шрифта

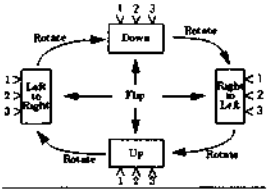


Рис. 4.26. Операции изменения положения блоков

Наконец, на рис. 4.27 наглядно показано действие ряда операций форматирования иного рода на вид простой модели, которая была описана чуть выше. Кроме того, на этом рисунке масштаб модели увеличен вдвое с помощью команды View ▸ Zoom In.

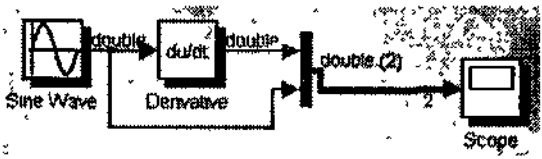


Рис. 4.27. Вид модели после операций форматирования

Этот рисунок демонстрирует возможности цветового оформления блоков, выделение нескалярных линий соединения, вывод данных о типе портов и введение указаний на порядок исполнения блоков в ходе моделирования.

Глава 5. Блоки источников и получателей сигналов

- Основная библиотека блоков
- Источники сигналов и воздействий
- Виртуальные регистраторы
- Библиотека Signal&Systems

Основная библиотека блоков

Версия пакета Simulink 4.0 имеет существенно обновленную и расширенную библиотеку блоков (компонентов). Она размещается в папке MATLAB/TOOLBOX/SIMULINK/BLOCKS. Основная палитра компонентов представлена файлом simulink.mdl. Как основная, так и дополнительные библиотеки Simulink представлены файлами разного формата: с расширением .dll, в виде mex-файлов и файлов с расширением .m. Последние могут при необходимости редактироваться и модифицироваться опытными пользователями.

Как уже отмечалось в главе 3, для вывода специального окна с разделами основной библиотеки Simulink (рис. 3 2) надо выделить эту библитеку в браузере библиотек и выполнить команду Open the "Simulink" Library контекстного меню. Каждая пиктограмма этого окна представляет группу компонентов определенного класса. Как видно из рис. 3.2, в этом окне содержатся следующие библиотеки:

- Sources — источники сигналов и воздействий;
- Sinks — регистрирующие устройства;
- Continuous — линейные компоненты;
- Discrete — дискретные компоненты;
- Math — математические компоненты;
- Functions&Tables — функции и таблицы;

- Nonlinear — нелинейные компоненты,
- Connections — соединительные компоненты;
- Signals&Systems — сигналы и системы;
- Blocksets&Toolboxes — дополнительные библиотеки и примеры;
- Demos — демонстрационные примеры пакета Simulink.

Источники сигналов и воздействий

Общий обзор источников

Строго говоря, окно Sources содержит пиктограммы источников воздействий. В электро- и радиотехнике их принято называть источниками сигналов, но в механике и в других областях науки и техники такое название не очень подходит — природа воздействий может быть самой разнообразной, например в виде перепада давления или температуры, механического перемещения, звуковой волны и т. д. Учитывая это, мы будем называть соответствующие компоненты просто источниками. Окно Sources представлено на рис. 5.1.

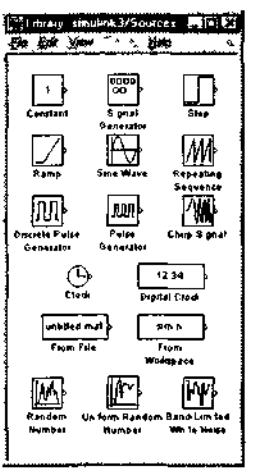


Рис. 5.1. Окно с перечнем источников сигналов и воздействий

Так, большинство компонентов представлено пиктограммами, изображающими временные или функциональные зависимости воздействий, например такие, как перепад для блока Step, синусоида для блока Sine Wave и т. д.

Набор блоков содержит практически все часто используемые при моделировании источники воздействий с самыми разными функциональными и временными зависимостями. Можно задать произвольное воздействие из файла — элемент From File. Имеются и случайные воздействия для моделирования систем и устройств методом Монте-Карло. С каждым блоком связано окно настроек, которое открывается при активизации пиктограммы блока в окне компонентов или в окне модели.

Для пользователей, имеющих хотя бы начальные представления о визуальном моделировании систем, установка параметров компонентов не вызывает трудностей, поскольку названия большинства параметров вполне очевидны. Например, для блока Sine Wave устанавливаются амплитуда синусоиды, частота, фаза и эталонное время (период дискретизации модели), то есть задаются типичные параметры сигнала (воздействия).

Для нового блока в окне параметров отображаются значения по умолчанию. Как правило, они нормализованы — например, заданы единичная частота, единичная амплитуда, нулевая фаза и т. д. Возможность изменения параметров появляется после переноса блока в окно модели. Как правило, значения параметров блоков по умолчанию позволяют начать моделирование и уточняются в ходе работы.

Источник постоянного воздействия Constant

Источник постоянного воздействия Constant задает константу. Рисунок 5.2 иллюстрирует применение этого источника и контроль уровня его воздействия с помощью осциллографа и цифрового индикатора Display. Подобные способы контроля мы будем применять и при описании других источников.

Источник постоянного воздействия характеризуется единственным параметром — своим уровнем воздействия в виде константы (по умолчанию 1) При установленном флажке Interpret vector parameters as 1-D вектор параметров интерпретируется как одномерный. Следует отметить, что можно задавать вектор констант в квадратных скобках: например, запись [-0.5, 1.0, 1.5] задает вектор из трех констант со значениями -0.5, 1.0 и 1.5. Установка флажка Interpret vector parameters as 1-D имеется и у других источников, и в дальнейшем мы ее специально отмечать не будем.

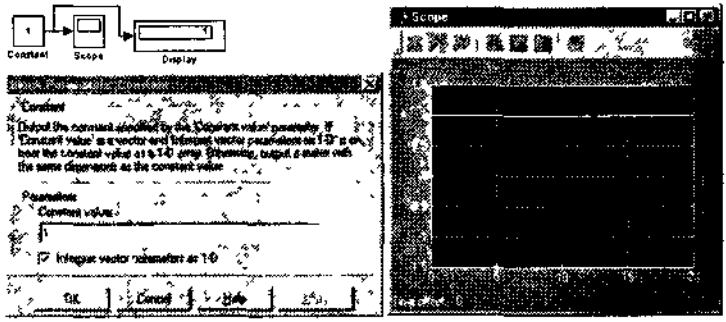


Рис. 5.2. Источник постоянного воздействия

Источник синусоидального воздействия Sine Wave

Синусоидальное воздействие — одно из самых распространенных. Мы уже описывали этот источник в главе 4 — см. рис. 4.5, на котором показано и окно параметров этого источника. Источник характеризуется амплитудой Amplitude, частотой Frequence, фазой Phase и эталонным временем Sample time. Последнее используется для согласования работы источника и других компонентов модели во времени (по умолчанию оно принято равным 0).

Источник нарастающего воздействия Ramp

Источник линейно нарастающего воздействия вида $F(t)=k*t$ и окно его параметров представлены на рис. 5.3. Осциллограф, подключенный к выходу источника, показывает временную зависимость этого воздействия.

Параметры источника:

- Slope — угловой коэффициент временной зависимости k;
- Start time — время, начиная с которого воздействие нарастает;
- Initial value — начальный уровень воздействия.

Как правило, указываются значения параметров, принятые по умолчанию.

Источник одиночного перепада Step

Источник воздействия в виде одиночного перепада показан на рис. 5.4.

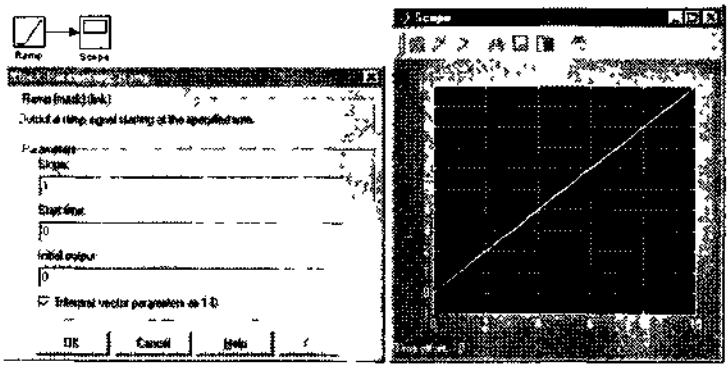


Рис. 5.3. Источник линейно нарастающего воздействия

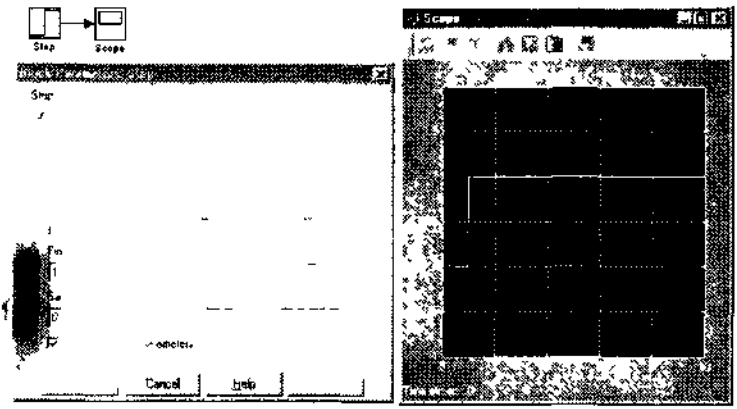


Рис. 5.4. Источник одиночного перепада

Параметры источника:

- Step time — время появления перепада (скачка);
- Initial value — начальное значение воздействия (до перепада);
- Final value — конечное значение воздействия (после перепада);
- Sample time — эталонное время.

Обратите внимание на то, что перепад можно задавать как положительным, так и отрицательным. Для задания отрицательного перепада начальное значение должно быть больше, чем конечное. Эти значения могут быть как положительными, так и отрицательными.

Сигнал-генератор Signal Generator

Источник воздействия типа сигнал-генератора служит для создания одного из четырех типов сигналов:

- sine — синусоидальный сигнал;
- square — прямоугольный периодический сигнал;
- sawtooth — пилообразный периодический сигнал;
- random — случайный сигнал.

Рисунок 5.5 показывает действие этого источника при моделировании пилообразного сигнала и окно его параметров. Выбор формы сигнала осуществляется в списке Wave form.

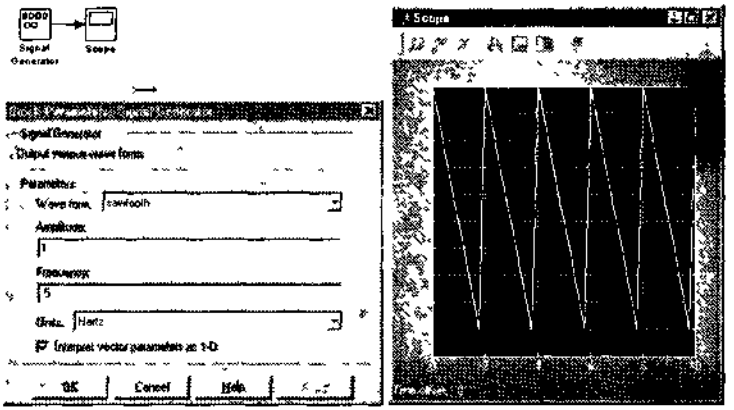


Рис. 5.5. Источник типа сигнал-генератора

Источник характеризуется двумя параметрами — амплитудой сигнала и его частотой. Следует отметить, что все создаваемые этим источником формы сигналов можно получить при помощи специализированных источников, описанных ниже.

Источник случайного сигнала с равномерным распределением Uniform Random Number

Источник случайного сигнала Uniform Random Number служит для генерации случайного сигнала с равномерным распределением. Уровень сигнала ограничен сверху и снизу значениями Maximum и Mi-

nimum. Применение этого источника и окно установки его параметров представлены на рис. 5.6.

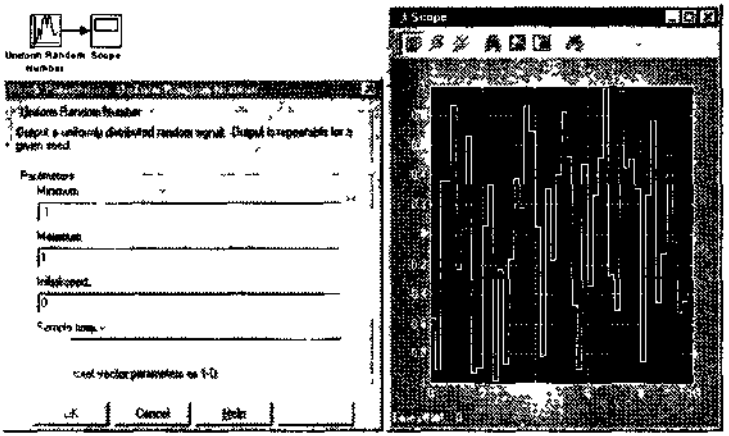


Рис. 5.6. Источник случайного сигнала с равномерным распределением

Осциллограмма сигнала также представлена на рис. 5.6. Надо помнить, что она не будет повторяться в ваших экспериментах, поскольку сигнал источника является случайным.

Источник случайного сигнала с нормальным распределением Random Number

Источник случайного сигнала Random Number служит для создания случайного сигнала с равномерным распределением уровня. Применение источника иллюстрирует рис. 5.7

Специфическими параметрами этого источника являются среднее значение сигнала Mean и среднеквадратическое отклонение Variance. Назначение двух других параметров (см. окно установки параметров на рис. 5.7) уже отмечалось.

Источник дискретных импульсов Discrete Pulse Generator

Источник дискретных импульсов Discrete Pulse Generator служит для создания прямоугольных импульсов (рис. 5.8).

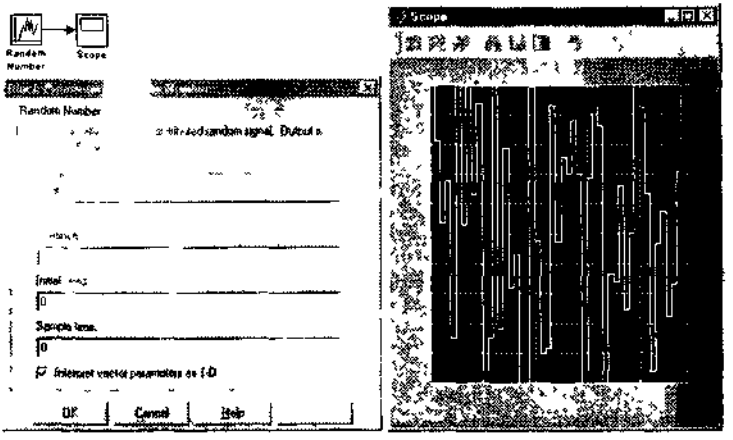


Рис. 5.7. Источник случайного сигнала с нормальным распределением

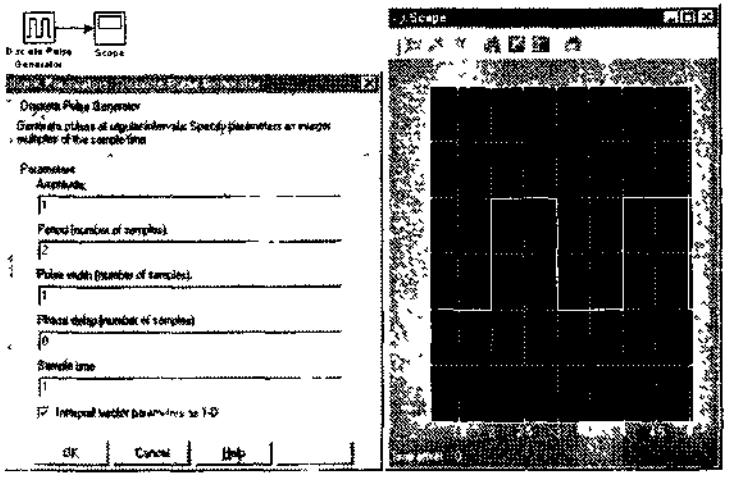


Рис. 5.8. Источник прямоугольных импульсов

Параметры источника по умолчанию обеспечивают формирование однополярных прямоугольных импульсов — см. осциллограмму на рис. 5.8. Можно устанавливать следующие параметры сигнала этого источника:

- Amplitude — амплитуда;
- Period — период (кратный эталонному времени);
- Pulse width — ширина импульсов (кратная эталонному времени);

- Phase delay — фазовая задержка (кратная эталонному времени);
- Simple time — эталонное время.

Генератор нарастающей частоты Chirp Generator

Генератор нарастающей частоты Chirp Generator создает почти синусоидальные колебания, частота которых увеличивается до некоторого момента времени (рис. 5.9).

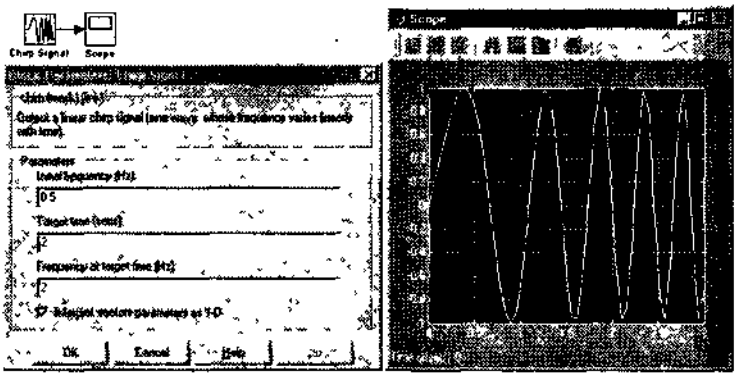


Рис. 5.9. Генератор нарастающей частоты

Параметрами генератора являются:

- Initial frequence — начальная частота в Гц (по умолчанию 0.1 Гц);
- Target time — время нарастания частоты (по умолчанию 100 с);
- Frequence at target time — конечное значение частоты в Гц (по умолчанию 1 Гц).

Обратите внимание, что в примере на рис. 5.9 заданы параметры, отличные от принятых по умолчанию.

Генератор белого шума Band Limited White Noice

Генератор белого шума Band Limited White Noice служит для создания шумового сигнала с заданной мощностью, равномерно распределенной по частоте. Рисунок 5.10 показывает работу этого генератора и окно установки его параметров.

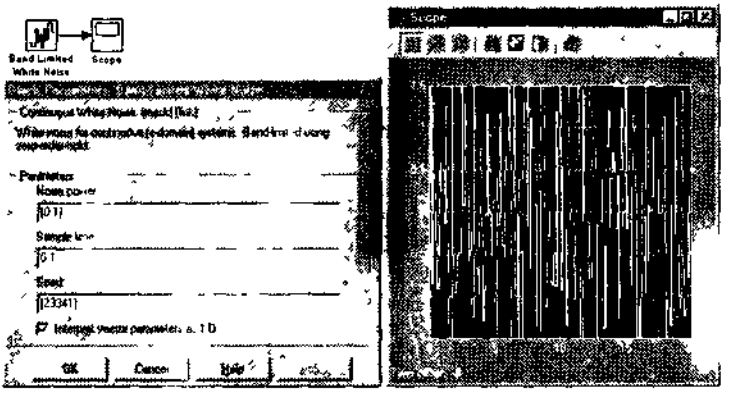


Рис. 5.10. Генератор белого шума

Генератор характеризуется мощностью шума Noice Power, эталонным временем Sample time и числом Seed, служащим для инициализации генератора случайных чисел. Генератор фактически является квантователем непрерывного сигнала, представляющего белый шум.

Источник времени моделирования Clock

Источник текущего времени Clock служит для генерации чисел, которые являются значениями текущего времени моделирования. Для контроля этого времени может использоваться цифровой индикатор — Display (рис. 5.11).

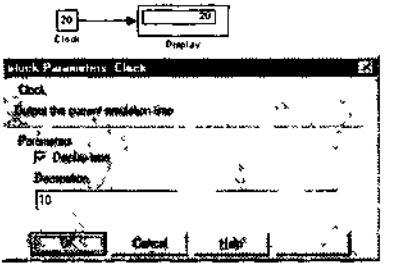


Рис. 5.11. Источник времени моделирования

Параметром источника является шаг Decimation, с которым меняются отсчеты времени. Флажок Display time задает отображение времени в блоке источника.

Цифровой источник времени Digital Clock

Имеется источник Digital Clock, имитирующий работу цифровых часов. Он имеет единственный параметр — эталонное время Sample time (по умолчанию 1 с). На рис. 5.12 показана работа источника Digital Clock.

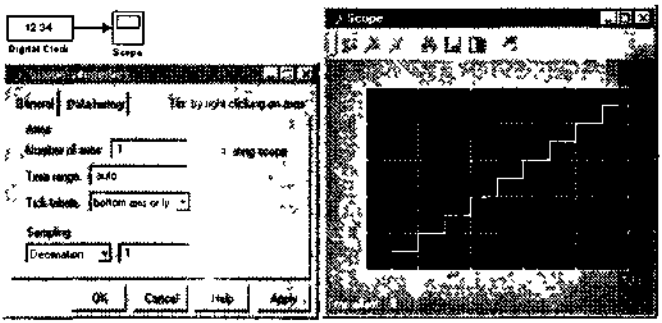


Рис. 5.12. Цифровой источник времени моделирования

Нетрудно заметить, что при заданном по умолчанию эталонном времени источник создает линейно нарастающий ступенчатый сигнал, что соответствует такому же росту текущего времени.

Блок From File

Блок From File служит для получения данных из внешнего файла. Данные должны быть представлены в виде матрицы:

$$\begin{bmatrix} t_1 & t_2 & t_{final} \\ u1_1 & u2_1 & u1_{final} \\ \\ un_1 & un_2 & un_{final} \end{bmatrix} \qquad (5.1)$$

Первая строка матрицы представляет собой идущие в возрастающем порядке отсчеты времени, а остальные строки — данные в эти моменты времени. В окне установки параметров следует задать имя файла, содержащего эту матрицу, и эталонное время Sample time. Выходной сигнал представляет собой только данные, то есть строка отсчетов времени в нем отсутствует.

Блок From Workspace

Блок From Workspace служит для получения данных из рабочего пространства в том же формате, который был описан для блока From

File. В качестве параметров задается формат матрицы данных Data (по умолчанию [T, U]) и эталонное время Sample time (по умолчанию 0). Кроме того, имеются флажки:

- Interpolate data — интерполяция/экстраполяция данных;
- Hold final data value — задержка последнего значения данных, что бывает нужно для его представления регистрирующими блоками.

При включении флажка Interpolate data производится интерполяция данных на промежутке времени до t_{final}, а для значений времени, больших t_{final}, выполняется экстраполяция по последним отсчетам данных (при снятии флажка сигнал на указанных интервалах обнуляется). Обратите внимание на то, что в отличие от матрицы источника From File в матрице источника From Workspace отсчеты времени занимают первый столбец, а не строку.

Для записи отсчетов времени и данных в файл и в рабочее пространство служат соответственно блоки To File и To Workspace. Они описаны в следующем разделе.

Виртуальные регистраторы

Обзор виртуальных регистраторов

Библиотека Sinks содержит блоки получателей информации. Их правильнее назвать регистрирующими компонентами. Наличие регистрирующих компонентов — важный фактор качественной визуализации результатов моделирования. На рис. 5.13 показано окно раздела библиотеки пакета Simulink с блоками виртуальных регистраторов.

Каждый регистратор имеет свое окно настройки, которое появляется при активизации его пиктограммы в окне компонентов или в окне модели.

В состав виртуальных регистраторов входят:

- Scope — осциллограф для наблюдения временных и иных зависимостей;
- XY Graph — графопостроитель в системе полярных координат;
- Display — устройство вывода на экран дисплея;
- To File — устройство записи данных в файл;

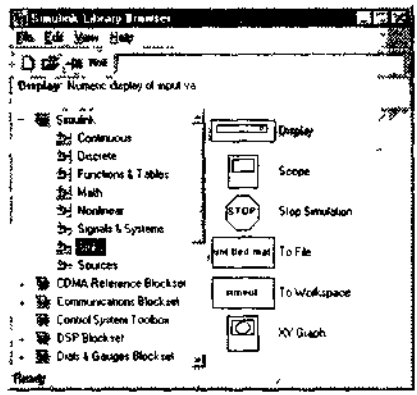


Рис. 5.13. Окно регистраторов пакета Simulink

- To Workspace — устройство записи в переменную рабочего пространства;
- Stop Simulation — блок остановки моделирования.

Важно отметить, что виртуальные регистраторы фиксируют параметры любого типа, а не только электрические. Это придает некоторым виртуальным регистраторам (приборам) уникальный характер. Например, об осциллографе, фиксирующем не только электрические сигналы, но и перемещения механических объектов, изменения температуры или давления и вообще изменения любых физических величин, даже крупные физические лаборатории могут только мечтать.

Виртуальный осциллограф

Виртуальный осциллограф, пожалуй, самое важное из регистрирующих устройств. Он позволяет представить результаты моделирования в виде временных диаграмм тех или иных процессов в форме, напоминающей осциллограммы современного высокоточного осциллографа с оцифрованной масштабной сеткой (и к тому же лучами разного цвета). Мы уже многократно приводили примеры применения осциллографа, например для контроля формы сигналов различных источников.

На рис. 5.12 показано окно параметров осциллографа с открытой вкладкой General, содержащей основные параметры:

- Number of axes — число осей (каналов) осциллографа;
- Time range — пределы временного интервала;

- Tick labels — вывод/скрытие отметок по осям;
- Sampling — установка временных соотношений: Decimation (в десятичных долях времени со значением по умолчанию 1) или Sample Time (в тактах эталонного времени, по умолчанию 0).

Параметр Number of axes позволяет превратить одноканальный осциллограф в многоканальный. При этом осциллограф приобретает несколько входных портов, к которым можно подключать различные сигналы. Пример применения осциллографа в таком режиме представлен на рис. 5.14.

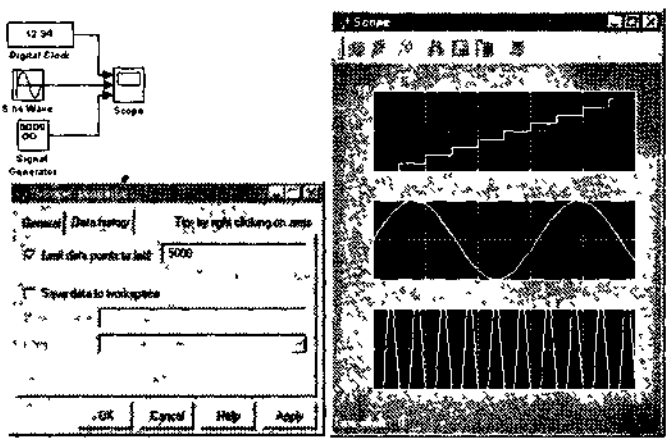


Рис. 5.14. Пример работы осциллографа в трехканальном варианте

На рис. 5.14 показано также окно параметров осциллографа с открытой вкладкой Data history. Здесь можно задать максимальное число точек осциллограмм для хранения и задать параметры хранения осциллограмм в рабочем пространстве системы MATLAB.

Назначение кнопок панели инструментов виртуального осциллографа представлено на рис. 5 15. Здесь особое внимание надо обратить на кнопки масштабирования, позволяющие (наряду с командами контекстного меню) менять размер осциллограммы. Весьма удобной является кнопка автоматического масштабирования — обычно она позволяет установить такой масштаб, при котором изображение осциллограммы имеет максимально возможный размер по вертикали и отражает весь временной интервал моделирования.

Рис. 5.15. Назначение кнопок панели инструментов окна осциллографа

Реальные осциллографы обычно имеют вход не только по вертикальной, но и по горизонтальной оси. В описанном виртуальном осциллографе такой вход не предусмотрен, но в этом и нет необходимости — подобную функцию имеет виртуальный графопостроитель, описываемый далее.

Виртуальный графопостроитель XY Graph

Графопостроитель — второе по распространенности устройство после осциллографа. В отличие от осциллографа виртуальный графопостроитель имеет входы по осям X и Y, что позволяет строить графики функций в полярной системе координат, фигуры Лиссажу, фазовые портреты и т. д. — см. рис. 5.16.

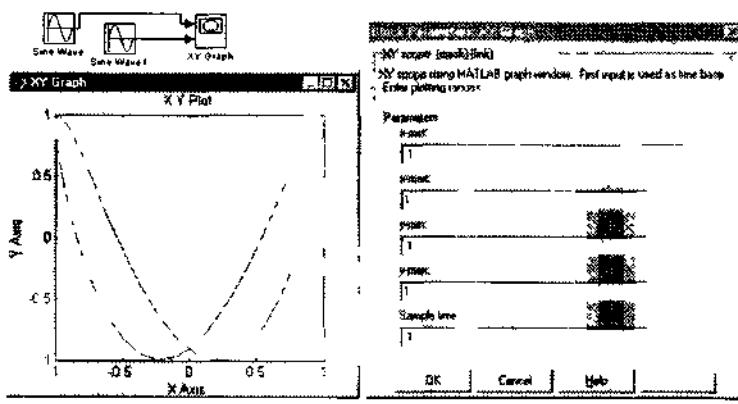


Рис. 5.16. Применение виртуального графопостроителя

На рис. 5.16 представлен пример построения фигур Лиссажу — они образуются при подаче на вход графопостроителя двух синусоидальных сигналов с частотами 1 и 2 Гц. На рисунке показано также окно

параметров графопостроителя. Эти параметры задают масштаб представления фигуры по осям X и Y и эталонное время для синхронизации с другими устройствами.

Дисплей Display

Виртуальный дисплей — устройство представления цифровой информации. Это одно из самых простых устройств. На рис. 5.17 дан пример применения дисплея для отображения значений вектора констант [0.5, 1.0, 1.5]. Чтобы все компоненты вектора были видны, дисплей в окне Simulink растянут мышью по горизонтали (его можно растягивать и по вертикали).

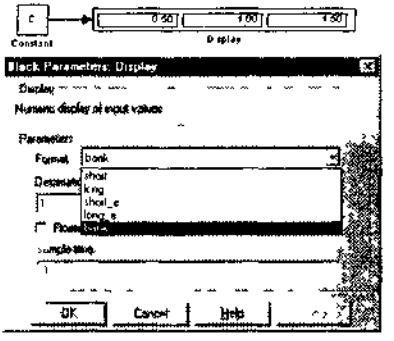


Рис. 5.17. Применение дисплея

Окно параметров дисплея также показано на рис. 5.17. В нем устанавливается формат отображения данных Format, представление входных данных в десятичных долях времени Decimation и в тактах эталонного времени Sample time. Можно также выбрать представление вещественных данных в формате с плавающей точкой (флажок Floating point). Дисплей обеспечивает динамическое отображение данных, то есть можно наблюдать их изменение в процессе моделирования.

Блок остановки моделирования Stop

Блок остановки моделирования обеспечивает прерывание моделирования и его остановку, если на его входе действует сигнал, не равный нулю. Работу блока поясняет рис. 5.18. На вход блока подан сигнал со значением 1. Поэтому, несмотря на установку времени мо-

делирования inf (бесконечность), осциллограф не фиксирует время — моделирование прерывается в самом начале.

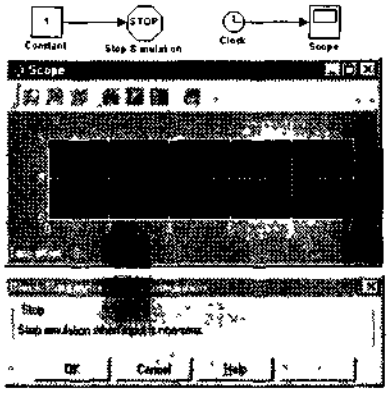


Рис. 5.18. Применение блока остановки моделирования

Для остановки моделирования в заданный момент времени нужно сформировать в этот момент отличный от нуля сигнал, а в остальное время обеспечить нулевой сигнал на входе блока Stop. Никаких параметров в этом блоке нет — его окно параметров (см. рис. 5.18) чисто информационное.

Блоки сохранения To File и To Workspace

Блоки сохранения To File и To Workspace записывают входные данные в виде матриц. Форматы матриц определяются так же, как и для блоков From File и From Workspace, сответственно.

На каждом такте моделирования формируется колонка, содержащая время такта и входные данные. Блок To File записывает полученную матрицу в файл с указанным именем (по умолчанию untitled.mat). Окно параметров блока показано на рис. 5.19.

В этом окне задается имя файла, максимальное число строк, разрядность в десятичных долях времени и эталонное время (чтобы зафиксировать состояние системы при t = 0, это время по умолчанию задается равным –1).

Блок To Workspace записывает указанную матрицу (но без строки отсчетов времени) в рабочее пространство. Окно его параметров представлено на рис 5.20.

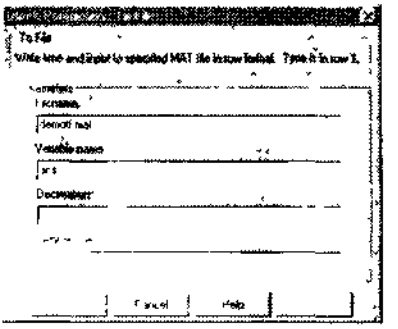


Рис. 5.19. Окно параметров блока To File

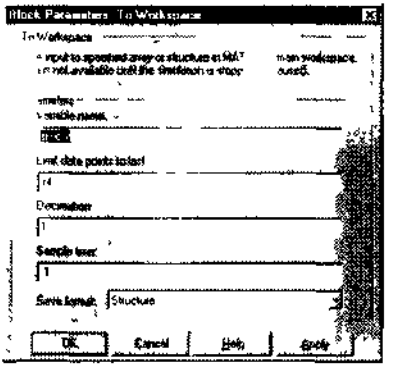


Рис. 5.20. Окно параметров блока To Workspace

Здесь помимо уже описанных параметров задается формат записи: структура Structure, структура со временем Structure with time и массив Array.

Напоминаем, что для использования данных из файла или рабочего пространства служат рассмотренные выше блоки From File и From Workspace.

ПРИМЕЧАНИЕ

Блоки To File и To Workspace применяются довольно редко, в основном для решения задач управления внешними устройствами в реальном масштабе времени с целью обмена с данными. В дальнейшем эти блоки использоваться не будут.

Библиотека Signal&Systems

Обзор библиотеки Signal&Systems

Окно библиотеки Signal&Systems представлено на рис. 5.21. Это одна из самых больших библиотек пакета Simulink. Однако она содержит в основном достаточно простые, по крайней мере на первый взгляд, блоки.

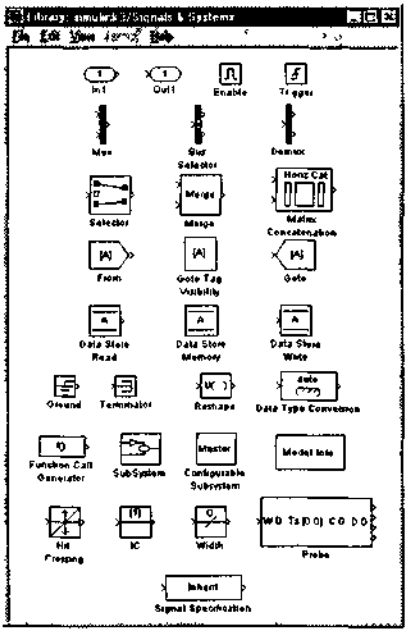


Рис. 5.21. Окно с блоками библиотеки Signal&Systems

Библиотека содержит следующие блоки:

- Data Store Memory — запись данных с заданным именем в память;
- Data Store Read — считывание данных с заданным именем;
- Data Store Write — запись значений сигналов с заданным именем;
- Data type conversion — преобразование данных к заданному типу, выбранному из списка;
- Enable — задание блоку признака управляемого блока;

- From — прием сигнала от блока Goto;
- IC — задание начального условия (инициализация);
- Goto — организация «беспроводного» передатчика данных;
- Ground — подключение к общему проводу (для входных портов);
- Hit Crossing — определяет момент времени, когда входной сигнал пересекает заданное значение;
- Master — доступ к мастеру подготовки подсистем;
- Model Info — вывод окна для задания информации о модели;
- Mux — мультиплексирование ряда входов;
- SubSystem — открытие пустого окна Simulink для создания подсистемы;
- Terminator — отвод для неиспользуемых (например, на этапе отладки модели) выходных портов;
- In — входной порт подсистемы для приема данных, поступающих извне в подсистему,
- Out — выходной порт подсистемы для передачи ее данных;
- Probe — контроль характеристик сигналов (длительности, типа и шага моделирования);
- Width — сигнал с уровнем, равным количеству входных сигналов.

Блоки Data Store Memory, Data Store Write и Data Store Read

Данные в моделях имеют различную судьбу. Некоторые данные «доживают» до конца моделирования, тогда как другие могут появиться только на короткое время, чтобы изменить состояние того или иного блока системы. Simulink имеет специальные средства для запоминания нужных данных и сохранения их до конца моделирования Они реализованы первыми тремя блоками.

Эти блоки позволяют работать с поименованными данными. Можно записывать (Data Store Write) и считывать (Data Store Read) поименованные данные. Блок Data Store Memory служит для записи поименованных данных в память, то есть резервирует под данные память. Установки параметров этого блока наиболее важны. Рисунок 5.22 показывает окно параметров этого блока.

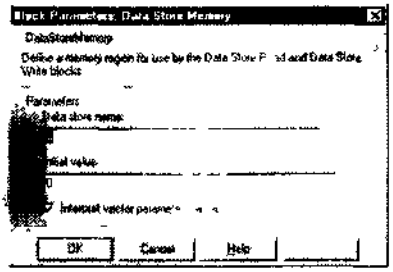


Рис. 5.22. Окно параметров блока Data Store Memory

В этом блоке предусмотрена установка трех параметров:

- Data store name — задает имя сохраняемых данных (например, A для записи переменной A),
- Initial value — задает начальное значение (по умолчанию 0) и формат сохраняемых данных (например, размер матрицы);
- Interpret vector parameters as 1-D — флажок, задающий интерпретацию вектора параметров данных как одномерного вектора.

Все три указанных блока используются всегда только совместно. При использовании блоков Data Store Write и Data Store Read обязательно надо учитывать тот формат данных, который задан параметрами блока Data Store Memory. Окна установки параметров этих блоков содержат два параметра: Data store name и Sample time. Параметр Interpret vector parameters as 1-D в этих блоках не задается, поскольку он наследуется из блока Data Store Memory.

Блок Data Store Memory может обслуживать несколько блоков Data Store Write и Data Store Read. Его расположением в диаграмме модели задается область видимости сохраняемых данных (в частности, переменных). Если блок Data Store Memory размещен в диаграмме верхнего уровня, то все ее данные будут «видны», то есть доступны для сохранения или считывания. Если же блок Data Store Memory расположен в субмодели, то доступными будут данные только этой субмодели.

Блоки Ground, Mux и Width

Блоки Ground, Mux и Width имеют настолько очевидное назначение, что не нуждаются в отдельном описании. Их действие демонстрирует рис. 5 23. Обратите внимание на то, что соединение, подключенное к земле, эквивалентно соединению с константой с нулевым значением. Цифровой индикатор показывает, что на выходе блока Mux в данном

случае два сигнала. В окне параметров блока Mux (рис. 5.23) устанавливается число входов и один из трех вариантов изображения блока.

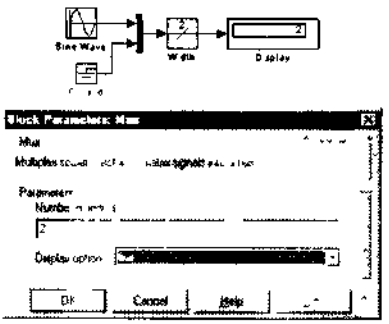


Рис. 5.23. Применение блоков Ground, Mux и Width

Блоки From, Goto и Goto Tag Visibility

Блоки From (Принять), Goto (Передать) и Goto Tag Visibility (Передать с учетом видимости) служат для организации обмена данными между блоками S-модели с учетом видимости данных. Они применяются для упрощения построения моделей.

С помощью блока Goto можно создать «беспроводной» передатчик данных, совместимых с этим блоком и имеющих имя (обычно в виде буквы). Один или ряд блоков From с таким же именем могут принимать данные от блока Goto, хотя прямых соединений между ними и блоком Goto нет.

Блок Goto Tag Visibility служит для установки правил видимости данных, подобных тем, что существуют в языках программирования при обмене данными между программой и отдельными процедурами. Эти блоки будут рассмотрены в главе 9 (как и блоки для создания подсистем)

Блок объединения сигналов в матрицу Matrix Concatenation

Блок объединения сигналов в матрицу Matrix Concatenation образует из ряда векторов матрицу, обеспечивая их объединение (конкатенацию) по горизонтали или по вертикали — рис. 5.24.

Параметрами блока являются число векторов и тип конкатенации — по вертикали или по горизонтали.

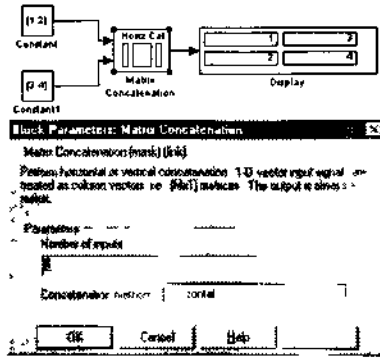


Рис. 5.24. Пример объединения векторов в матрицу

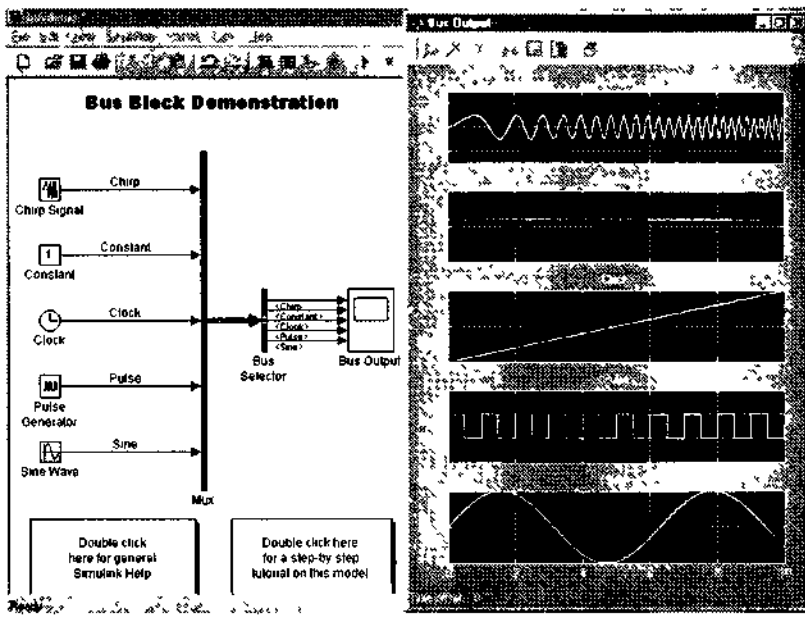


Рис. 5.25. Пример объединения и селекции сигналов ряда источников

Блок шинного селектора Bus Selector

Блок шинного селектора Bus Selector обеспечивает выбор заданных сигналов из нескольких. В примере, приведенном на рис. 5.25, сигналы ряда источников объединяются в шину с помощью блока Mux, а затем разделяются с помощью блока шинного селектора, после чего

каждый сигнал подается на отдельный вход многоканального виртуального осциллографа

Окно параметров шинного селектора представлено на рис 5 26 Окно состоит из двух частей в левой имеется список всех входных сигналов, а в правой — выбранных (селектированных) сигналов В данном случае выбраны все сигналы, но можно выбрать любые

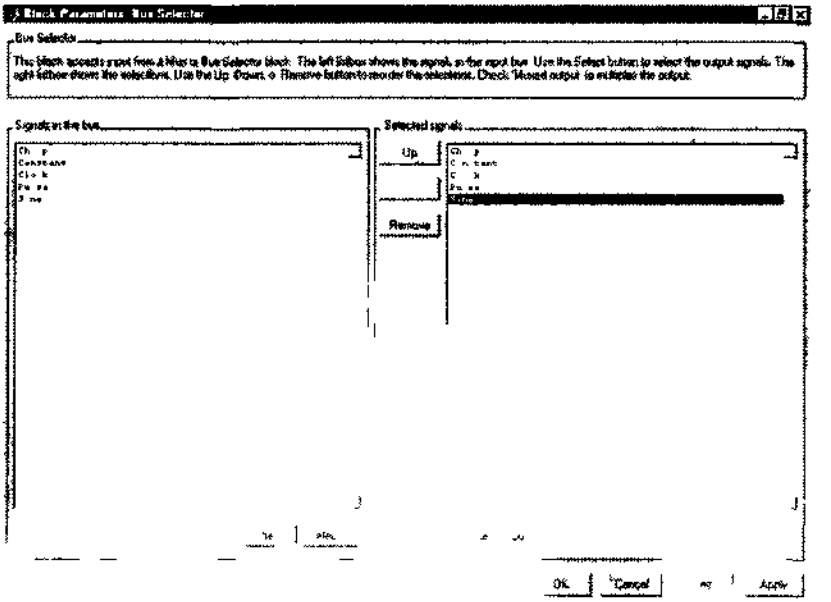


Рис. 5 26 Окно установки параметров шинного селектора

Для выбора того или иного сигнала надо выделить его в левом окне и нажать кнопку Select>> Флажок Muxed output позволяет наблюдать описание сигналов на выходе селектора Кнопка Up позволяет переносить вверх выделенный в правой части окна сигнал и таким образом менять порядок сигналов Кнопка Remove удаляет выделенный сигнал из окна выбранных, а кнопка Refresh очищает окно выбранных сигналов

Блок спецификации сигнала Signal Specification

Блок спецификации сигнала Signal Specification служит для распознавания сигнала на его входе и выдачи спецификации Пример при-

менения блока дан на рис 5 27 В окне параметров блока устанавливается размерность сигнала Dimension, эталонное время Sample time, тип данных Data Type и тип сигнала Signal type

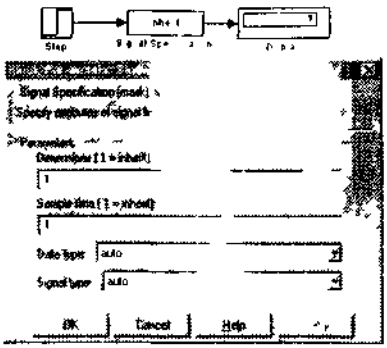


Рис. 5.27. Применение блока Signal Specification

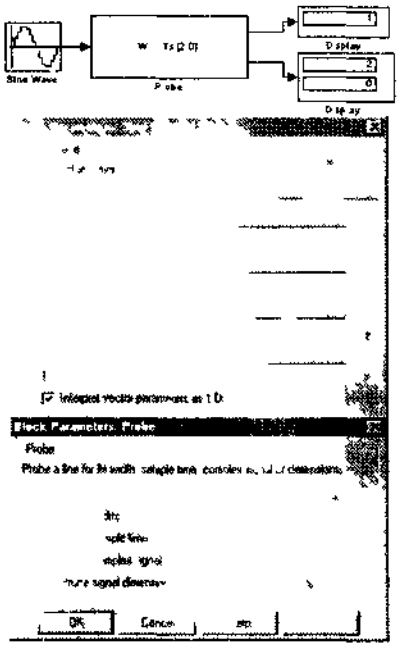


Рис. 5.28. Применение блока Signal Specification

Блок проверки сигналов Probe

Блок Probe осуществляет более детальную проверку сигналов, чем блок Signal Specification (рис. 5.28). На этом рисунке представлены также окна параметров источника синусоидального сигнала и блока Probe. При возникновении ошибки с выходов блока Probe считываются те значения, которые указаны в окнах установки параметров. Эти значения можно использовать для оценки ситуаций, вызывающих ошибки, и устранения ошибок в ходе моделирования.

Этот блок имеет следующие параметры:

- Probe width — проверка длительности сигнала;
- Probe Simple time — проверка эталонного времени;
- Probe Complex Signal — проверка сигнала на принадлежность к комплексным числам (возвращает 1, если сигнал представлен в комплексном виде, и 0 в противном случае);
- Probe signal dimension — проверка размерности сигнала.

Контролируются те параметры, для которых установлены флажки. Числом отмеченных флажков задается число выходов блока.

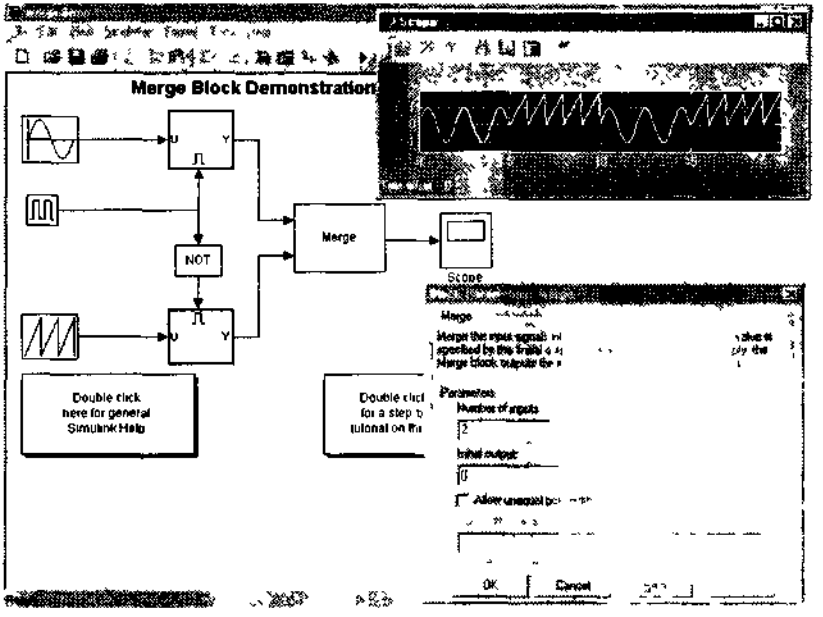


Рис. 5.29. Применение блока Merge

Блок выбора последнего сигнала Merge

Блок Merge передает на выход то значение, которое было получено на входе последним. Пример применения этого блока и окно его параметров представлены на рис. 5.29.

В этом примере на вход осциллографа поочередно подается сигнал то от источника синусоидального сигнала, то от источника линейно изменяющегося сигнала. В окне параметров блока Merge задается число входов и параметр инициализации выхода (по умолчанию пустой список, означающий, что на выход передается входной сигнал, значение которого было вычислено последним)

Глава 6. Математические блоки

- Математическая библиотека Math
- Непрерывные блоки
- Блоки функций и таблиц

Математическая библиотека Math

Обзор библиотеки Math

Большие возможности в моделировании различных систем предоставляет библиотека математических блоков Math. Она впервые появилась в версии пакета Simulink 3.1. Окно библиотеки Math показано на рис. 6.1. Хотя это окно дано в формате окон Simulink 3, более удобном для обзора библиотек с большим числом блоков, блоки в нем относятся к версии Simulink 4.

Возможность задания разнообразных математических блоков с настраиваемыми свойствами имеет большое значение для выполнения прозрачного для пользователя математического моделирования как простых, так и сложных устройств и систем. В этом заключается одно из главных достоинств пакета Simulink.

Блоки выполнения арифметических операций

К числу наиболее простых математических блоков относятся блоки арифметических операций: вычисления абсолютного значения числа Abs, скалярного произведения Dot Product, обычного произведения Product, а также суммы Sum. Рисунок 6.2 показывает применение трех таких блоков. Там же даны окна установки параметров этих блоков. Для контроля работы блоков к их входам подключаются константы, а к выходам — цифровой регистратор (дисплей).

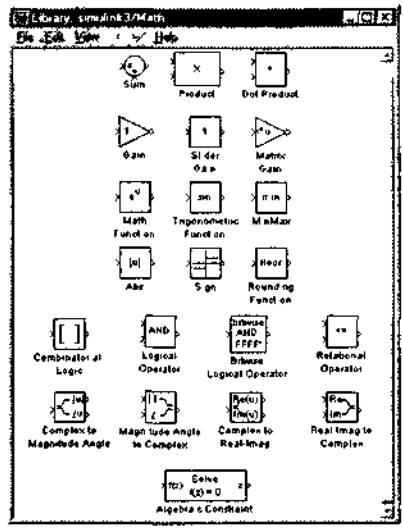


Рис. 6.1. Окно библиотеки математических блоков

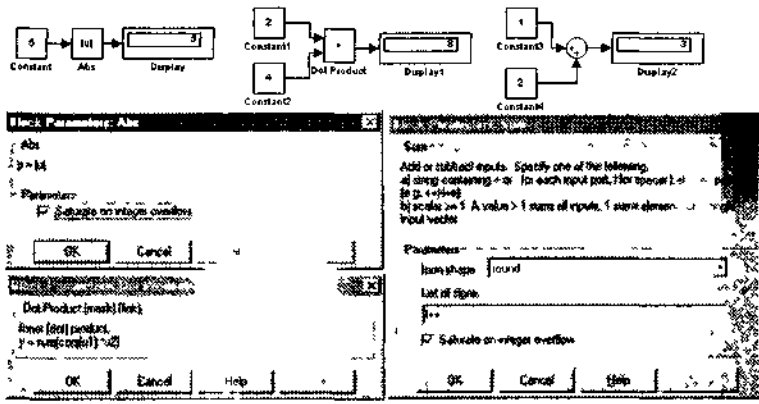


Рис. 6.2. Примеры применения блоков арифметических операций

Обратите внимание на то, что в окне настройки блока сложения/вычитания можно установить вид представления блока (круглый или квадратный) и число входов с выполняемыми по ним операциями. Число входов и операции задаются шаблоном List of sign. Например, шаблон |++ означает, что блок имеет два суммирующих входа, а |+-+ — что он имеет три входа, причем средний — вычитающий, а крайние — суммирующие.

Блок Product (Умножение) предназначен для умножения и деления ряда входных сигналов. При этом операции задаются подобно тому, как это было описано для блока суммирования/вычитания с применением знаков умножения * или деления / в шаблоне.

Для контроля знака служит блок Sign. Он возвращает –1 при отрицательном входном аргументе, 0 — при нулевом входном аргументе и 1 — *при положительном входном аргументе*

Блоки вычисления элементарных функций

На рис. 6.3 представлены три блока, выполняющих вычисления математических функций: Math Function (Математическая функция), Trigonometric Function (Тригонометрическая функция) и Rounding Function (Функция округления).

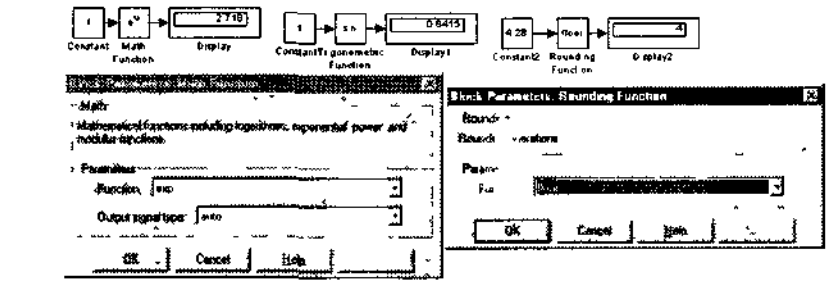


Рис. 6.3. Примеры применения блоков арифметических операций

Действие этих функций и установки их параметров достаточно очевидны. Выбор конкретной вычисляемой функции осуществляется в раскрывающемся списке. В нем имеется типовой набор элементарных функций.

Блок логических операций Logical Operation

Этот блок позволяет задавать любую из известных базовых логических операций. Пример выполнения операции AND представлен на рис. 6.4. Пример формирует таблицу истинности для этой операции.

На рис. 6.4 представлено также окно установки параметров логического блока. Параметрами являются тип блока (выбирается из

списка) и число входов. Нетрудно заметить, что могут быть заданы следующие логические операции:

- AND — логическое умножение (операция И);
- OR — логическое сложение (операция ИЛИ);
- NAND — операция НЕ-И;
- NOR — операция НЕ-ИЛИ;
- XOR — операция сложения по модулю 2 (исключающее ИЛИ);
- NOT — операция логического отрицания (НЕ).

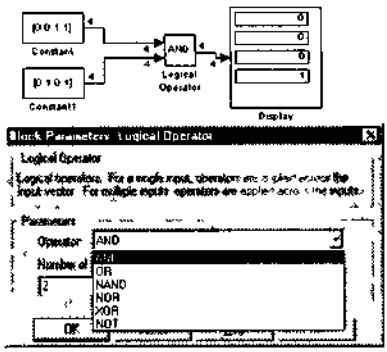


Рис. 6.4. Пример выполнения операции AND

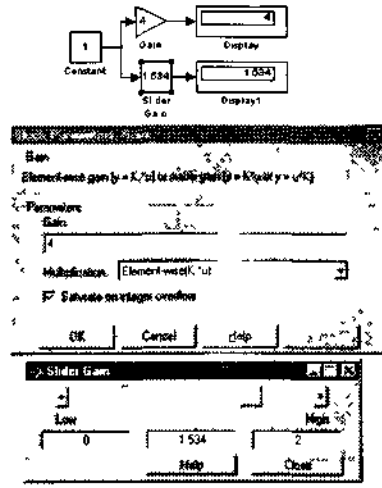


Рис. 6.5. Примеры применения блоков масштабирования

Обратите внимание, что в приведенном примере входными данными являются векторы с логическими сигналами — логическим нулем 0 и логической единицей 1. Для побитовых логических операций (довольно редких и специфических) служит блок Bitwise Logical Operation.

Блоки масштабирования Gain и Slider Gain

Для масштабирования данных (умножения их на заданный коэффициент — константу) служат блоки Gain и Slider Gain (рис. 6.5). В блоке Gain константа вводится в окне параметров (по умолчанию 1), а в блоке Slider Gain ее можно выбирать с помощью ползунка.

Для масштабирования матричных данных служит блок Matrix Gain.

Блоки Complex to Magniitude-Angle и Complex to Real-Imag

Блоки обработки комплексных данных Complex to Magniitude-Angle и Complex to Real-Imag служат для вычисления абсолютного значения и фазы комплексного числа и выделения из него действительной и мнимой частей. Действие блоков представлено на рис 6 6

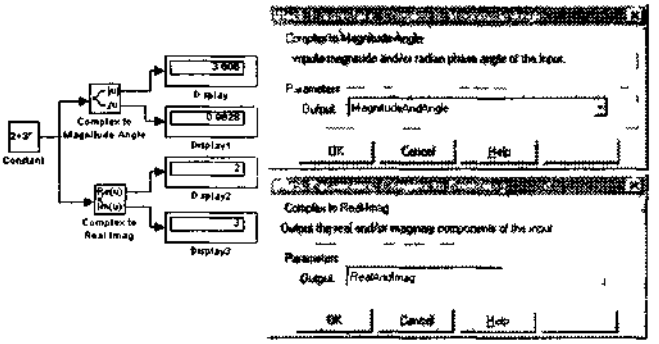


Рис. 6.6. Примеры применения блоков обработки комплексных данных

Можно вычислять либо любой из выходных параметров отдельно, либо оба одновременно

Блок поиска минимума и максимума MinMax

Для поиска в данных минимального и максимального значений служит блок MinMax (рис. 6 7).

Для выбора выходного параметра (минимума или максимума) служит раскрывающийся список Function в окне установки параметров.

Блок алгебраического ограничения Algebraic Constraint

Блок алгебраического ограничения Algebraic Constraint служит для вычисления значений переменных исходя из заданных (обычно в виде уравнения или системы уравнений) ограничений Иными словами, этот блок служит для решения систем уравнений, накладывающих ограничения на значения переменных (рис. 6 8).

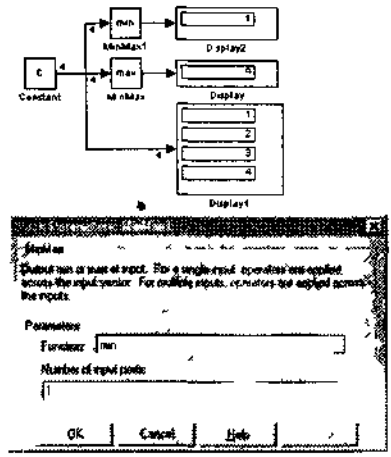


Рис. 6.7. Примеры применения блоков масштабирования

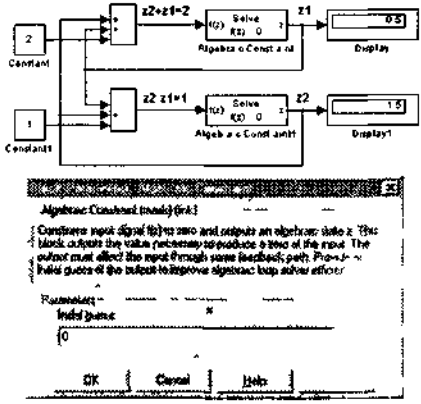


Рис. 6.8. Пример решения системы из двух уравнений

Модель рис. 6 8 является примером решения типовой математической задачи средствами моделирования пакета Simulink Здесь с помощью блоков суммирования/вычитания формируются два линейных уравнения Правые их части представлены блоками задания констант 2 (для верхнего уравнения) и 1 (для нижнего уравнения). Для совместного решения составленных таким образом уравнений и служат блоки Algebraic Constraint. Полученное решение выводится с помощью блоков дисплеев

Непрерывные блоки

Непрерывные (Continuous) блоки также играют важную роль в создании математических моделей многих устройств. Достаточно отметить электрические фильтры, построенные на таких компонентах (например, на операционных усилителях), широко используемые в технике электро- и радиосвязи, или математические блоки, применяемые в аналоговых ЭВМ. На рис. 6.9 представлен раздел библиотеки Continuous с непрерывными компонентами.

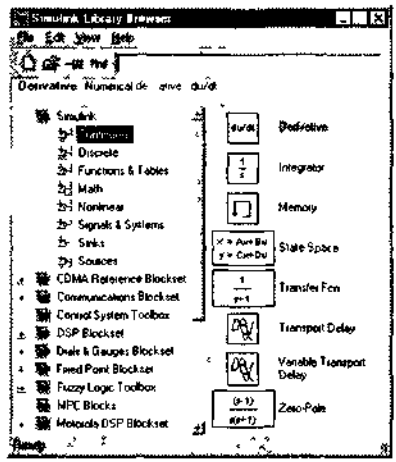


Рис. 6.9. Раздел библиотеки с непрерывными блоками

Среди блоков этой библиотеки особое значение имеют блоки дифференцирования и интегрирования, выполняющие широко распространенные операции высшей математики, а также блоки временной задержки и решения уравнений и систем уравнений.

Дифференцирующий блок Derivative

Дифференцирующий блок Derivative служит для численного дифференцирования входных данных (сигналов). На рис. 6.10 дан пример последовательного интегрирования и дифференцирования пилообразных импульсов, а также окно установки параметров дифференцирующего блока.

Нетрудно заметить, что никаких параметров блок дифференцирования не имеет и его окно установки параметров дает только ин-

формацию о назначении блока. Из осциллограмм рис. 6.10 можно заметить, что как интегрирование (см. ниже), так и дифференцирование выполняются не идеально, но, тем не менее, форма пилообразного сигнала после последовательного выполнения этих операций практически восстанавливается.

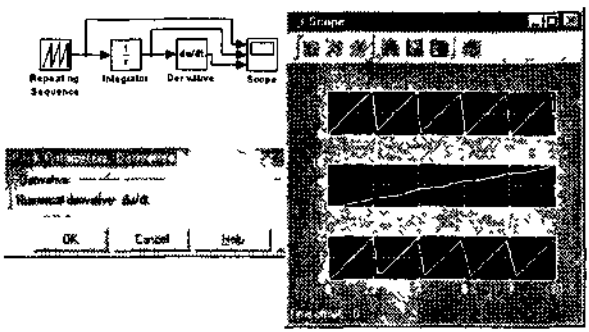


Рис. 6.10. Интегрирование пилообразных импульсов с последующим дифференцированием

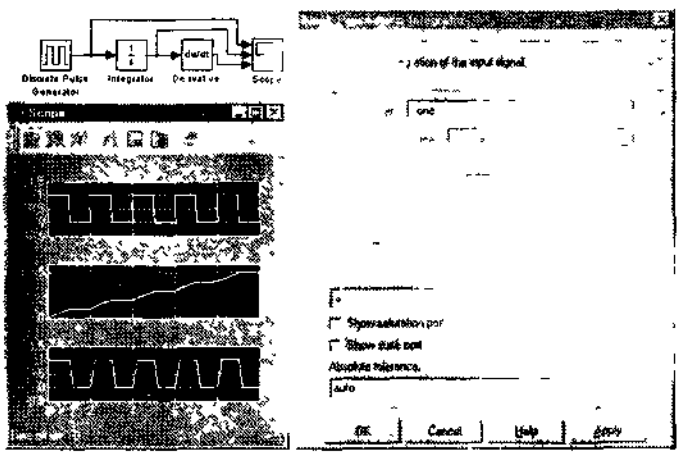


Рис. 6.11. Интегрирование прямоугольных импульсов с последующим дифференцированием

Интегрирующий блок Integrator

Блок интегрирования Integrator выполняет функции интегрирования входных данных (сигналов). На рис. 6.11 приведен еще один

пример совместного применения блоков интегрирования и дифференцирования — для входного сигнала в виде прямоугольных импульсов. На рис. 6.11 показано также окно установки параметров интегрирующего блока.

Окно параметров интегрирующего блока содержит следующие элементы:

- External reset (внешний сброс) — тип внешнего управляющего сигнала, выбираемый из раскрывающегося списка (none — нет, rising — нарастающий, falling — спадающий, either — любой);
- Initial condition source — источник начального значения выходного сигнала при интегрировании. В раскрывающемся списке можно выбрать внутренний (internal) или внешний (external) источник;
- Initial condition (начальное состояние) — установка начального значения выходного сигнала при интегрировании (в виде числа, по умолчанию 0);
- Limit output — включение/отключение ограничения выходного сигнала;
- Upper saturation limit — верхний предел ограничения выходного сигнала (по умолчанию inf, то есть +?);
- Lower saturation limit — нижний предел ограничения выходного сигнала (по умолчанию -inf, то есть -?);
- Show saturation port — управляет отображением порта, выводящего уровни ограничения выходного сигнала;
- Show state port — управляет отображением порта состояния системы;
- Absolute tolerance — абсолютная погрешность (по умолчанию автоматический выбор — auto).

Блок Memory

Блок Memory (Память) запоминает входной сигнал и смещает его на один такт времени (рис. 6.12).

В окне параметров этого блока устанавливается параметр Initial condition — начальное состояние (по умолчанию 0). Если флажок Inherit sample time установлен, то берется шаг изменения времени, равный эталонному времени Sample time предшествующего блока.

Если этот флажок сброшен, то используется шаг, равный 0.1 модельного времени.

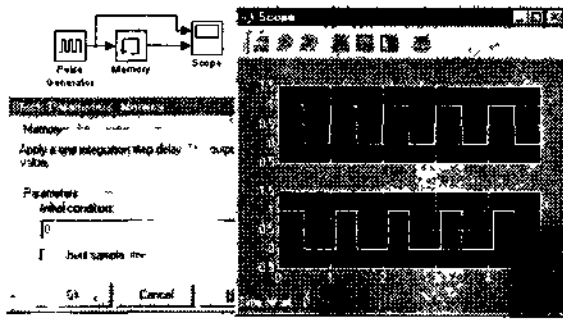


Рис. 6.12. Применение блока Memory

Блок фиксированной задержки Transport Delay

Блок фиксированной задержки Transport Delay обеспечивает временную задержку входного сигнала на заданное время (рис. 6.13).

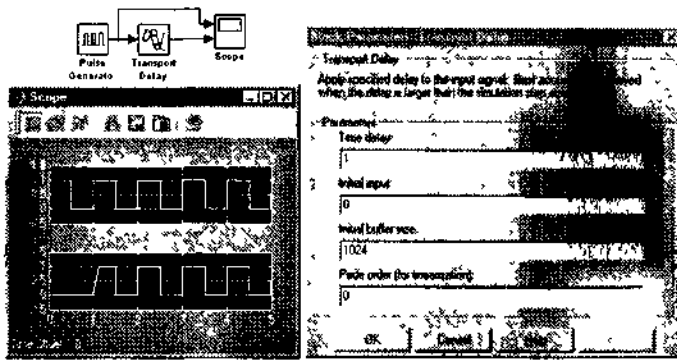


Рис. 6.13. Применение блока Transport Delay

Параметры блока:

- Time Delay — время задержки (по умолчанию 1);
- Initial input — начальный уровень входа (по умолчанию 0);
- Buffer size — размер буфера, выделяемого под задержанный сигнал, в байтах (число, кратное 8, по умолчанию 1024 байт);

- Pade order (for linearization) — порядок линеаризации Паде (по умолчанию 0, но может задаваться как целое положительное число для повышения точности линеаризации).

Полезно обратить внимание на то, что задержка может задаваться вещественным числом.

Блок управляемой задержки Variable Transport Delay

Блок управляемой задержки Variable Transport Delay имеет два входа: один для задерживаемого сигнала, а другой для сигнала управления. Он позволяет создавать задержку, заданную уровнем сигнала управления (рис. 6.14).

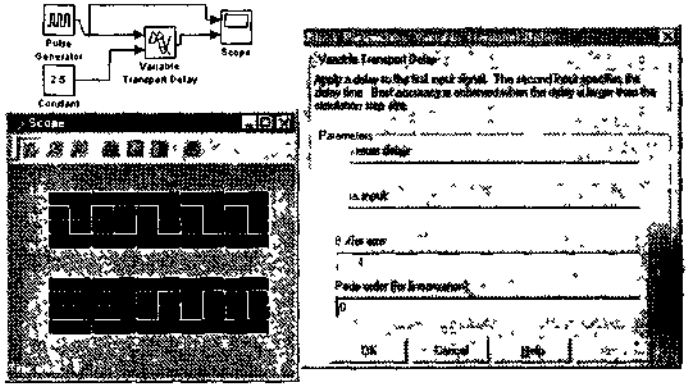


Рис. 6.14. Применение блока Variable Transport Delay

Параметры этого блока совершенно аналогичны уже описанным параметрам блока фиксированной задержки (за исключением того, что вместо параметра Time delay используется параметр Maximum delay — максимальная задержка). Пример на рис. 6.14 демонстрирует задержку на 2.5 такта эталонной частоты, что задается константой 2.5 на входе управления блока управляемой задержки.

Блок задания линеаризованной модели State-Space

В ходе моделирования Simulink автоматически составляет некоторую исходную модель системы, которая представляет собой систе-

му нелинейных алгебраических уравнений. Для вычисления изменения на каждом малом шаге эта система линеаризируется и приводится к матричной системе уравнений:

$$dx/dt = A\ x + B\ u$$
$$y = C\ x + D\ u$$

где x — вектор состояния, u — вектор входных воздействий и y — вектор выходных сигналов. После добавления к переменным их изменений создается новая система уравнений состояния, она вновь линеаризируется, выполняется новый шаг моделирования и т. д.

Блок State-Space позволяет задать линеаризованную матричную модель системы. Структура ее матриц представлена на рис. 6.15. Здесь m — число входов, n — число состояний системы, r — число выходов. Рисунок 6.15 дает представление о размерах матриц в линеаризованной матричной системе уравнений.

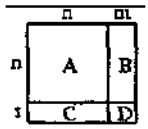


Рис. 6.15. Структура матриц линеаризованной матричной системы уравнений

На рис. 6.16 показано окно параметров блока State-Space. Основными параметрами здесь являются матричные коэффициенты, по умолчанию равные 1, и параметр Initial condition, задающий вектор начального состояния (по умолчанию 0).

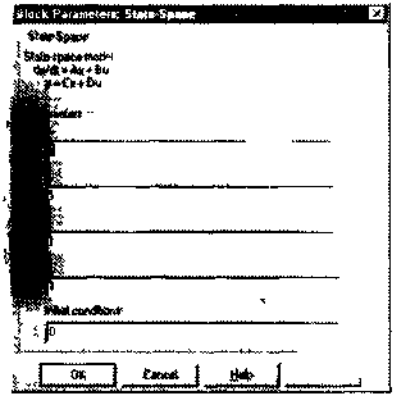


Рис. 6.16. Окно параметров блока State-Space

Для моделирования конкретных систем и устройств этот блок практически не применяется. Он служит, скорее, для задания некоторых абстрактных или теоретических систем, анализ которых выходит за рамки данного учебного курса.

Блок передаточной характеристики Transfer Fcn

Блок передаточной характеристики Transfer Fcn создает передаточную функцию H(s)=y(s)/u(s) в виде отношения полиномов заданной степени. Вид блока после задания его параметров представлен на рис. 6.17.

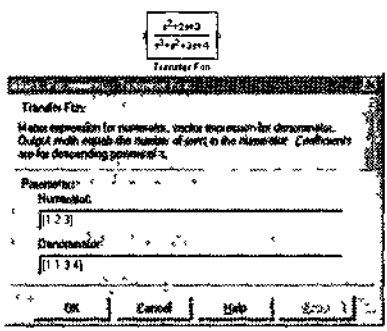


Рис. 6.17. Пример задания параметров блока Transfer Fcn

Блок Transfer Fcn имеет два параметра — векторы коэффициентов полиномов числителя Numerator и знаменателя Denominator. Они задают вид выражения H(s), которое и появляется внутри блока.

Блок Zero-Pole

Блок Zero-Pole служит для создания передаточной функции с заданными полюсами и нулями. Вид блока и его передаточная функция в общем виде представлены на рис. 6.18.

$$H(s) = K\frac{Z(s)}{P(x)} = K\frac{(s-Z(1))(s-Z(2))\ldots(s-Z(m))}{(s-P(1))(s-P(2))\ldots(s-P(n))}$$

Рис. 6.18. Блок Zero-Pole и его обобщенная передаточная функция H(s)

Окно параметров блока Zero-Pole представлено на рис. 6.19. В этом окне задаются списки нулей (поле Zeros) и полюсов (поле Poles) передаточной характеристики, а также коэффициент передачи (поле Gain).

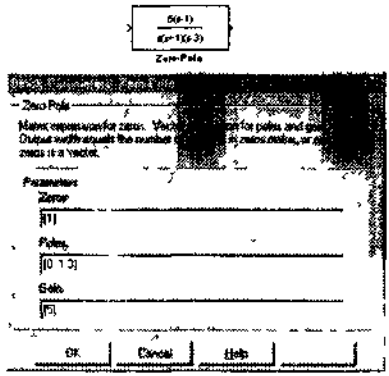


Рис. 6.19. Окно параметров блока Zero-Pole

Блоки Transfer Fcn и Zero-Pole находят довольно ограниченное применение, так что ограничимся приведенными данными о них.

Блок оператора отношения Relational Operator

Блок Relational Operator служит для реализации операции отношения между двумя сигналами. В отличие от блока Logical Operator, блок оператора отношения может работать со смешанными сигналами, например с вектором и скаляром. При этом значение каждого элемента вектора сравнивается со значением скалярного операнда. Результатом будет вектор, значения элементов которого соответствуют результатам сравнения (рис. 6.20).

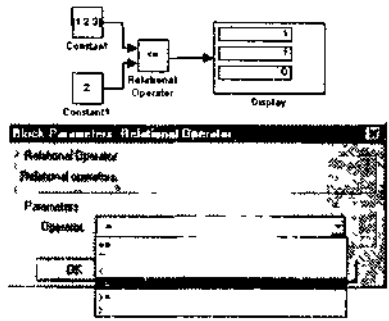


Рис. 6.20. Блок Relational Operator

Этот блок имеет единственный параметр — тип оператора (Operator). Он устанавливается в раскрывающемся списке — на рис. 6.20 этот

список показан открытым, и из него видно, какие операторы может реализовать данный блок.

Блок комбинаторной логики Combinatorical Logic

Блок Combinatorical Logic обеспечивает преобразование входного сигнала в соответствии с правилами, задаваемыми так называемой таблицей истинности. Эта таблица формируется по правилам, принятым в теории конечных автоматов. Опишем кратко эти правила.

Число строк в таблице истинности определяется соотношением

number of rows = 2 ^ (number of inputs),

где number of inputs — число входных сигналов. Индекс каждой строки (вектора u длиной m) определяется выражением:

row index = $1 + u_m + u_{m-1}*2^1 + .. + u_1*2^{(m-1)}$.

Ниже представлен пример формирования таблицы истинности для логической операции AND.

Строка (Row)	Вход 1 (Input 1)	Вход 2 (Input 2)	Выход (Output)
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Например, входной вектор [1 0] соответствует третьей строке, поскольку (2^1*1 + 1) = 3. В соответствии с логикой операции AND мы должны задать в третьей строке значение выходного сигнала 0. Более сложные примеры таблиц истинности (в том числе с векторами выходных сигналов) заинтересованный читатель найдет в постановке своих задач из области моделирования конечных автоматов.

Пример работы блока Combinatorical Logic при реализации логической операции AND приведен на рис. 6.21.

ВНИМАНИЕ

Блок Combinatorical Logic — очень мощное средство для решения задач логического моделирования. Успех решения определяется прежде всего умением правильно задавать таблицу истинности в соответствии с решаемой логической задачей. Необходимо помнить, что входными и выходными сигналами этого блока являются логические константы 0 (FALSE) и 1 (TRUE).

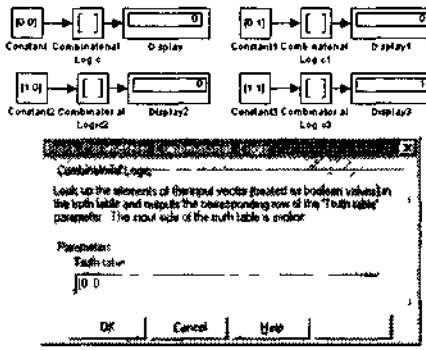


Рис. 6.21. Применение блока Combinatorical Logic для реализации операции AND

Обратите внимание на то, что в таблице истинности — единственном параметре этого блока — фигурируют только выходные сигналы. На рис 6.21 представлены все 4 возможных состояния блока. При работе с этим блоком для объединения ряда отдельных сигналов часто используется блок Mux.

Блоки функций и таблиц

Обзор блоков функций и таблиц

Еще один новый раздел библиотеки — Functions&Tables, содержащий компоненты функций и таблиц, представлен на рис. 6.22.

Как вытекает из наименования данного раздела, он представлен двумя категориями блоков — функциями и таблицами. Блоки функций позволяют вводить в модели практически любые функции, что имеет большое значение в математическом моделировании различных объектов и систем. Это выгодно отличает Simulink от многих систем моделирования (например, схемотехнических), в которых возможности задания математических функций отсутствуют или представлены слишком сложно.

Блоки таблиц обеспечивают задание табличных данных с различной размерностью, а также позволяют использовать линейную или сплайновую интерполяцию и экстраполяцию данных. Эти блоки позволяют эффективно вводить данные экспериментов и строить на их основе простые модели устройств и систем.

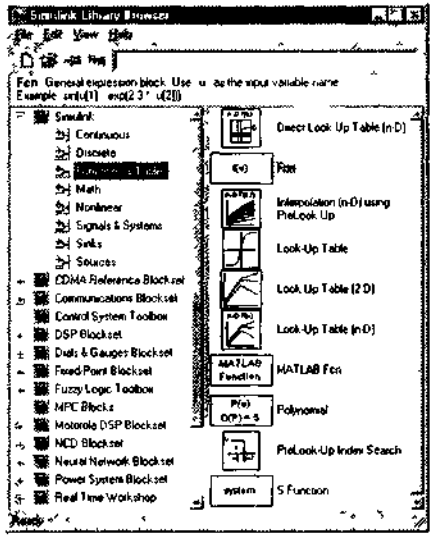


Рис. 6.22. Блоки функций и таблиц и окна их настройки

Учитывая важность средств этого раздела библиотеки, число блоков в нем для Simulink 4.0 увеличено вдвое по сравнению с предшествующей версией (с 5 блоков до 10). Среди новых блоков можно отметить многомерные таблицы, таблицы с прямым доступом и блок полиномиальных выражений.

Блок задания функции Fcn

Блок Fcn служит для задания функций одной переменной u или ряда переменных u(i) На рис. 6.23 показан пример применения блока Fcn для функции двух переменных u(1) и u(2), заданной по умолчанию. Входным сигналом блока может быть вектор с числом компонентов, равным числу переменных.

В окне параметров блока имеется поле Expression для ввода выражения, задающего нужную функцию. Это выражение составляется по правилам, принятым для описания функций на языке С. Перечислим допустимые операторы в порядке уменьшения приоритета их операций:

- круглые скобки ();
- унарные операторы – и +;
- оператор возведения в степень ^;
- оператор логического отрицания !;

- операторы арифметического умножения * и деления /;
- операторы арифметического сложения + и вычитания –;
- логические операторы отношения <, >, <= и >=;
- операторы отношения «равно» = и «не равно» !;
- оператор логического умножения && (И);
- оператор логического сложения || (ИЛИ).

Заметим, что операторы отношения и логические операторы возвращают логические значения в виде логического нуля (FALSE) или логической единицы (TRUE). В выражениях для функций этого типа могут использоваться и переменные системы MATLAB, находящиеся в рабочем пространстве, например переменная ans.

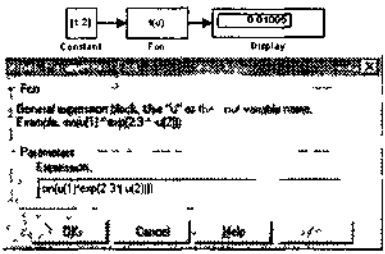


Рис. 6.23. Блок задания функции Fcn

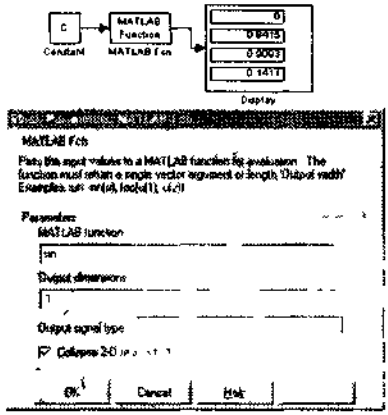


Рис. 6.24. Блок задания функции MATLAB Fcn

Блок задания функции MATLAB Fcn

Блок задания функции MATLAB Fcn служит для задания функций одной переменной u или ряда переменных u(1) по правилам, принятым для языка программирования базовой системы MATLAB 6.0. В частности, это означает, что в теле функции могут встречаться как встроенные функции системы MATLAB, так и любые процедуры и функции, реализованные в виде m-файлов.

Рисунок 6.24 демонстрирует применение блока MATLAB Fcn для вычисления функции синуса, заданной только ее именем — sin. При этом на входе задается вектор значений входного сигнала, а на выходе формируется вектор выходного сигнала с той же длиной.

Окно параметров этого блока (см. рис. 6.24) содержит описание правил задания функции и раздел параметров Parameters. В разделе MATLAB function задается выражение для функции и длина вектора выхода Output width. Если она должна совпадать с длиной вектора входного сигнала, то вводится значение –1.

Правила задания выражения для функции совпадают с правилами, рассмотренными для функции Fcn. Если выражение состоит из одной встроенной функции (см. пример на рис. 6.24), то достаточно задать имя функции без входных параметров.

В раскрывающемся списке Output signal type можно выбрать тип выходного сигнала в виде вещественного чила (real), комплексного (complex) или задать автоматический выбор (auto).

Блок полиномиальных выражений Polynomial

Блок Polynomial служит для задания степенного многочлена P(u) в виде вектора его коэффициентов. Они задаются в порядке уменьшения степени независимой переменной u. Например, для задания полинома x^2+2x+3 надо задать вектор коэффициентов полинома [1, 2, 3]. Рисунок 6.25 поясняет применение блока Polynomial: показан результат вычисления значений полинома x^2+2x+3 для входного вектора [0, 1, 2, 3].

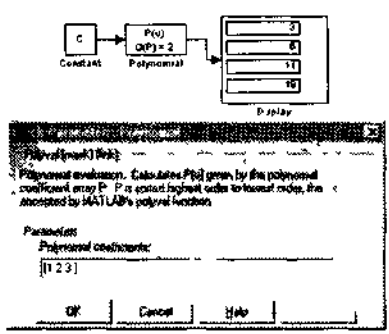


Рис. 6.25. Блок полиномиальных выражений Polynomial

В пиктограмме блока отображается старшая степень заданного полинома. В окне параметров блока задается только вектор его коэффициентов.

Блок одномерной таблицы Look-Up Table

Блок Look-Up Table служит для задания в табличной форме некоторых данных, представляющих ряд значений функции одной переменной. Входом блока является вектор значений входного сигнала, а выходом — вектор соответствующих значений выходного сигнала. На рис. 6.26 дан пример определения функции гиперболического тангенса для значений входного аргумента от –5 до 5 с шагом 1.

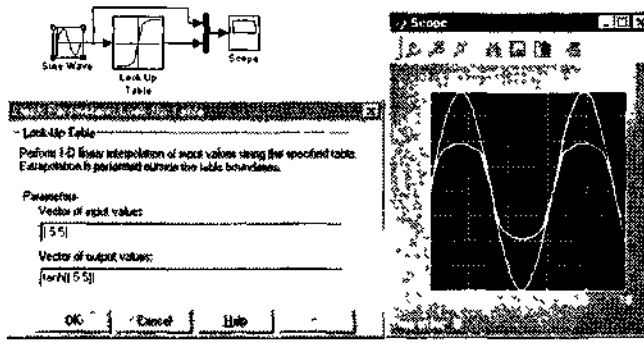


Рис. 6.26. Блок задания одномерной таблицы

Необходимо обратить внимание на то, что блок обеспечивает линейную интерполяцию и экстраполяцию для входных сигналов, уровни которых имеют промежуточные значения (находятся между узловыми точками таблицы) или даже выходят за пределы заданного интервала изменения аргумента. Окно параметров блока содержит описания входного и выходного векторов данных. Выходной вектор может содержать как явно заданные численные данные, так и записи функций от ранжированных переменных — в данном случае это функция tanh([-5:5]), имеющая 11 значений в интервале изменения аргумента от –5 до 5 с шагом 1.

Блок двумерной таблицы Look-Up Table (2D)

Блок Look-Up Table (2D) служит для задания таблиц, представляющих значения функции двух переменных. Это может быть, например, семейство линий (оно отображается на пиктограмме блока). Как и в предшествующем блоке, предусмотрена линейная интерполяция и экстраполяция входных данных. Рисунок 6.27 иллюстрирует применение этого блока.

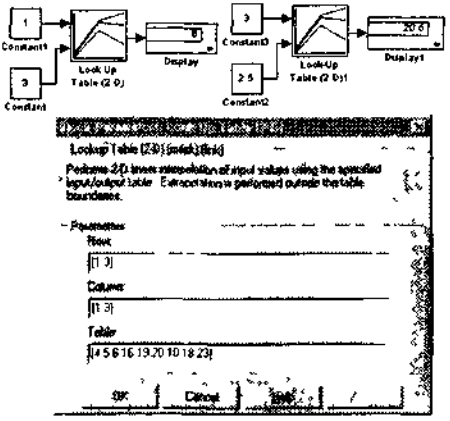


Рис. 6.27. Блок задания двумерной таблицы

В окне параметров блока двумерной таблицы задаются векторы индексов строк Row и столбцов Column, а также матрица данных таблицы Table.

Блок многомерной таблицы Look-Up Table (n-D)

Блок Look-Up Table (n-D) служит для табличного представления данных и имеет расширенные возможности в части задания размерностей таблиц и интерполяции и экстраполяции их данных.

Многомерные таблицы и их описания довольно громоздки и применяются редко, так что мы ограничимся примерами применения этого блока для случая двумерных таблиц, показанных на рис. 6.28.

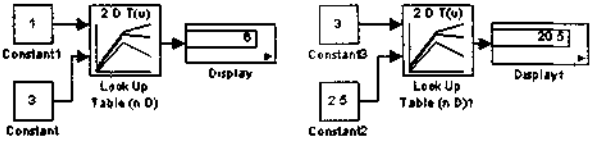


Рис. 6.28. Применение блока задания многомерных таблиц

Окно параметров этого блока показано на рис. 6.29. В связи с расширенными возможностями этого блока число его параметров достаточно велико в сравнении с ранее рассмотренными блоками.

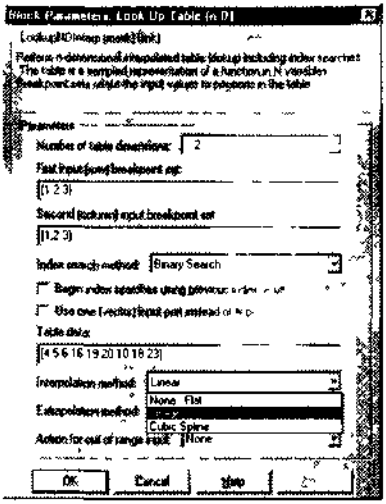


Рис. 6.29. Применение блока задания многомерных таблиц

Блок Look-Up Table (n-D) имеет следующие параметры:

- Number of table dimension — размерность таблицы;
- First input (row) breakpoint set — индексы строки входа;
- Second (column) input breakpoint set — индексы столбца входа;
- Index search method — метод просмотра индексов (Evenly Spaced Points — поиск по равномерно распределенным точкам, Linear Search — линейный поиск, Binary Search — бинарный поиск);
- Table data — данные таблицы (в общем случае многомерный массив);
- Interpolation method — выбор метода интерполяции (None-flat — без интерполяции, Linear — линейная интерполяция, Cubic spline — кубическая сплайновая интерполяция);
- Extrapolation method — выбор метода экстраполяции (None-flat — без экстраполяции, Linear — линейная экстраполяция, Cubic spline — кубическая сплайновая экстраполяция);
- Action for out of range input — действия при выходе значений входного сигнала за допустимые пределы.

Кроме того, в окне параметров имеются два флажка:

- Begin index searches using previous index results — начинать обзор индексов, используя предыдущие индексы;

- Use one (vector) input port instead of N points — использовать один входной порт (вектор) вместо N портов.

Наличие обширного набора настроек этого блока придает ему гибкость и универсальность

Блок Interpolation (n-D) using PreLoop-Up

Блок Interpolation (n-D) using PreLoop-Up позволяет создать многомерную интерполяционную таблицу для представления в табличном виде функций ряда переменных. Пример применения блока дан на рис. 6 30.

Окно параметров этого блока несколько проще, чем у предшествующего блока. Назначение параметров рассмотрено выше.

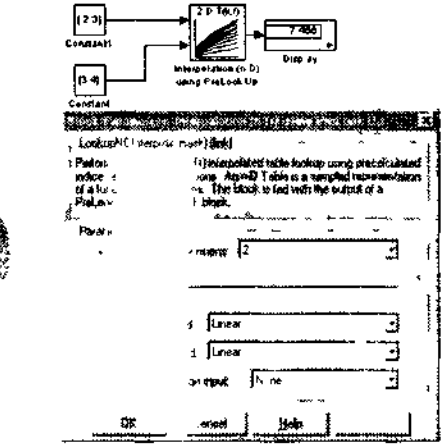


Рис. 6.30. Применение блока Interpolation (n-D) using PreLoop-Up

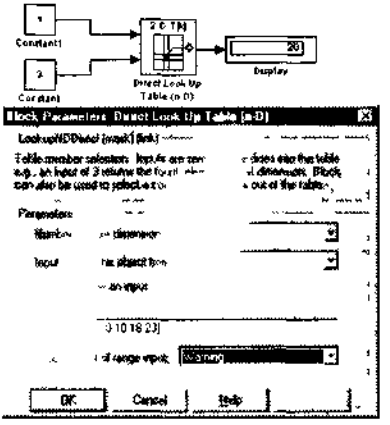


Рис. 6.31. Применение блока Direct Loop-Up Table (n-D)

Блок таблицы с прямым доступом Direct Loop-Up Table (n-D)

Блок Direct Loop-Up Table (n-D) позволяет задавать многомерные таблицы с прямым доступом к их элементам Отказ от интерполяции и экстраполяции существенно уменьшает время доступа, что бывает нужно, если интерполяция и экстраполяция осуществляются внешними блоками и по специальным алгоритмам. Пример применения этой таблицы дан на рис. 6.31.

Блок имеет следующие параметры:

- Number of table dimentions — размер таблицы;
- Input select this object from table — входная селекция объекта из таблицы (выбирается из раскрывающегося списка: Element — элемент, Column — столбец и 2-D Matrix — матрица размером 2 × 2);
- Make table an input — сделать таблицу входом;
- Table data — массив данных таблицы;
- Action for out of range input — действие при выходе входных данных за допустимые пределы: None — без сообщения, Warning — предупреждение, Error — сообщение об ошибке (по умолчанию — Warning).

Блок работы с индексами PreLook-Up Index Search

Блок PreLook-Up Index Search обеспечивает вычисление принадлежности одномерных данных к ближайшим узлам при приближении к ним снизу, а также контроль разности (дистанции) в относительных единицах между данными и значениями этих узловых точек. Данные могут быть представлены отдельными значениями или одномерным вектором значений — см. рис. 6.32, где показана обработка данных вектора из трех констант [12, 45, 78].

Для этого блока задаются следующие параметры:

- Breakpoint data — вектор (или ранжированная переменная) для узловых точек;
- Index search method — метод поиска индексов (Binary Search — бинарный поиск, Linear Search — линейный поиск или Evently Spaced Points — поиск по равномерно распределеным точкам);
- Process out of range input — тип процесса при выходе входного сигнала за заданные пределы (Clip to Range — ограничить по пределу или Linear Extrapolation — линейная экстраполяция);
- Action for out range input — действие при выходе входного сигнала за заданные пределы (None — отсутствие сообщения, Warning — предупреждающее сообщение, Error — сообщение об ошибке).

Имеются также два флажка:

- Begin index search using previous index result — начало просмотра индексов с последнего результата;
- Output only the index — вывод только индексов.

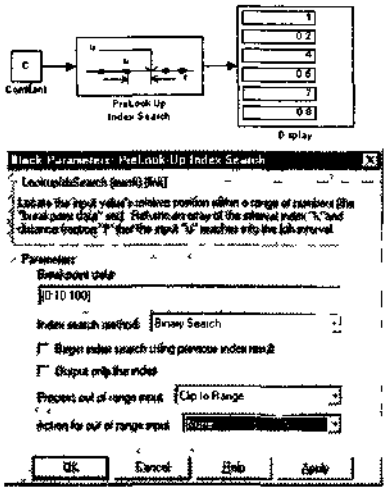


Рис. 6.32. Применение блока PreLook-Up Index Search

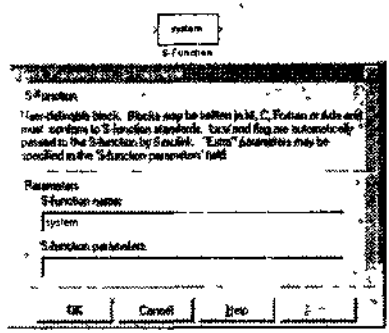


Рис. 6.33. Блок S-функций и окно его параметров

Блок задания S-функций

Блок задания S-функций одновременно самый простой и самый сложный блок. Простой, потому что окно его параметров, представленное на рис. 6.33, содержит лишь поля для ввода наименования S-функции и ее параметров.

Сложность S-функций обусловлена тем, что они являются вполне самостоятельными программами, написанными на языках C, Ада, Фортран и MATLAB, и представлены файлами соответствующих форматов. Основное назначение S-функций связано прежде всего с созданием новых блоков и решением нестандартных задач, которые трудно решаются с помощью имеющихся библиотек пакета Simulink.

Обычным пользователям трудно представить ситуацию, когда задачу нельзя решить с помощью стандартых библиотечных блоков. А потому возможности S-функций интересны лишь достаточно квалифицированным пользователем.

Примеры применения S-функций

Для получения доступа к демонстрационным примерам на применение S-функций достаточно выполнить в командном режиме MATLAB следующую команду:

```
>> sfundemos
```

Эта команда выводит окно с весьма внушительным набором демонстрационных примеров. Мы не приводим вид этого окна, поскольку прочесть названия примеров можно только на экране дисплея высокого качества.

Описание S-функций и их демонстрационных примеров явно выходит за рамки тематики и объема этой книги. Поэтому ограничимся демонстрацией лишь одного примера, представленного на рис. 6.34. Этот пример демонстрирует работу высотомера с применением блока S-функции, записанного на языке программирования Фортран.

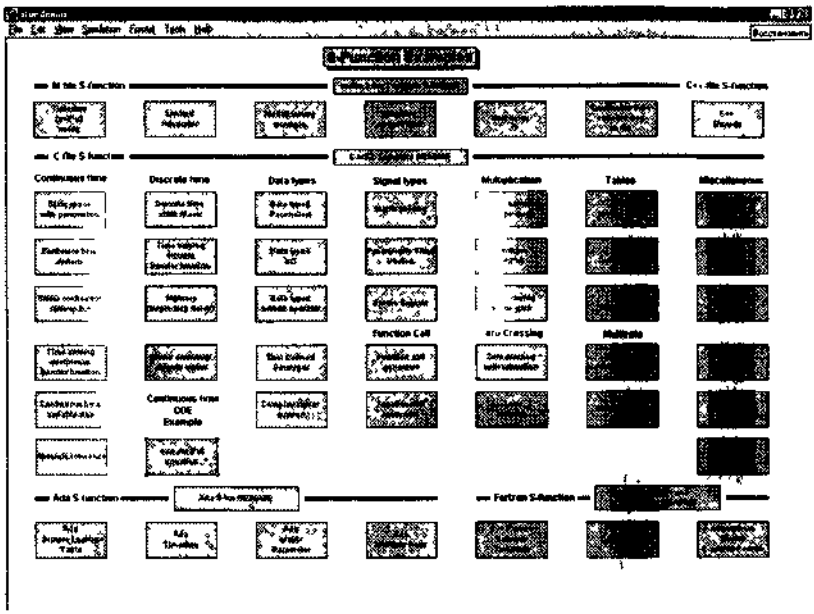


Рис. 6.34. Пример применения блока S-функции, записанной на языке Фортран

Из этого примера ясно, что применение S-функций в пакете Simulink так же просто, как применение и любого библиотечного модуля. Сложность заключается в достаточно серьезной и сложной работе по подготовке самих S-функций.

Глава 7. Нелинейные и дискретные блоки

- Нелинейные блоки
- Дискретные блоки

Нелинейные блоки

Обзор нелинейных блоков

Пакет Simulink предназначен главным образом для моделирования нелинейных динамических систем. Раздел Nonlinear основной библиотеки Simulink, посвященный нелинейным компонентам, содержит наиболее распространенные нелинейные блоки (рис. 7.1).

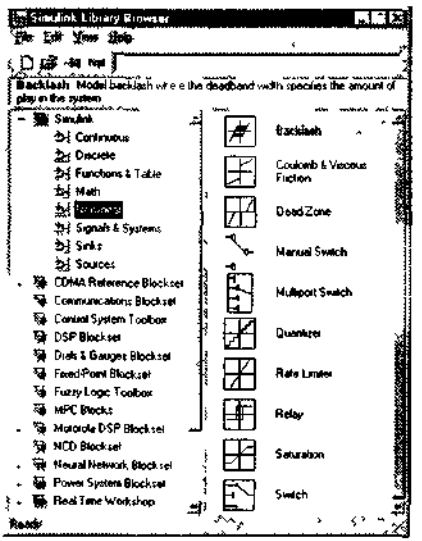


Рис. 7.1. Окно библиотеки с нелинейными компонентами

Среди нелинейных блоков следует отметить блоки с типичными нелинейностями, например блоки с характеристиками в виде ти-

повых математических функций, компоненты идеальных и неидеальных ограничителей и т. д. Достойно представлены и такие сложные компоненты, как квантователи сигналов, блоки нелинейности, моделирующие нелинейные петли гистерезиса, и ключи — переключатели с разными состояниями, зависящими от управляющих сигналов.

Важным параметром нелинейного устройства является его передаточная функция — зависимость выходного сигнала от входного, $y(u)$. Для некоторых блоков, например релейного или квантующего, они имеют разрывный характер. Передаточные характеристики указаны в описании, которое дается в окне параметров каждого нелинейного блока. В связи с этим в тексте соответствующих разделов формулы передаточных характеристик опущены.

ВНИМАНИЕ

Следует обратить внимание на то, что параметры нелинейных блоков могут задаваться не только численными значениями, но и списками и вычисляемыми выражениями. Большинство нелинейных блоков рассматриваются как идеальные в том смысле, что инерционность устройств, которые представляются такими блоками, не учитывается

Блок ограничения Saturation

Блок Saturation представляет собой нелинейное устройство, сигнал на выходе которого равен входному сигналу до тех пор, пока не достигает порогов ограничения: верхнего Upper limit или нижнего Lower limit. После этого сигнал перестает изменяться. Наиболее характерно применение ограничителя для ограничения синусоидальных сигналов (рис. 7.2).

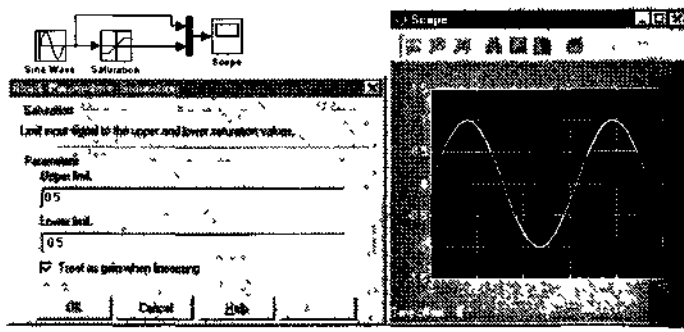


Рис. 7.2. Блок ограничения и окно установки его параметров

Как видно из рис. 7.2, окно параметров блока содержит лишь поля для установки верхнего (Upper limit) и нижнего (Lower limit) порогов ограничения.

Блок с зоной нечувствительности Dead Zone

Еще одна характерная нелинейность — линейная зависимость выходного сигнала от входного (с вычетом соответствующего порога) везде, за исключением зоны нечувствительности (мертвой зоны). Эта нелинейность моделируется блоком Dead Zone (рис. 7.3).

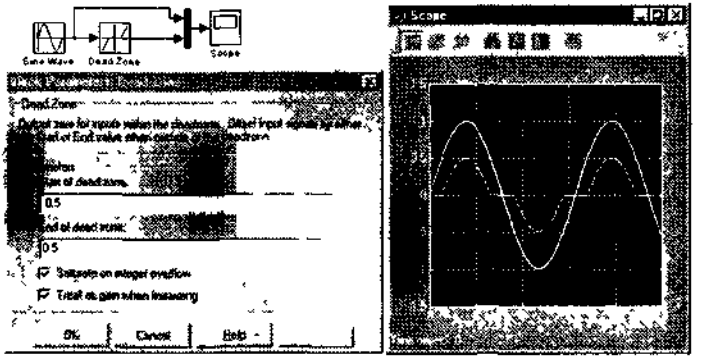


Рис. 7.3. Блок с зоной нечувствительности и окно установки его параметров

Окно параметров этого блока содержит границы зоны нечувствительности Start of dead zone и End of dead zone. По умолчанию они заданы равными −0.5 и 0.5. Флажки Saturate on integer owerflow (Ограничение при переполнении целых) и Treat as gain when linearizing (Трактовать как ограничения при линеаризации) по умолчанию включены

Релейный блок Relay

Релейный блок Relay имеет разрывную передаточную функцию с гистерезисом (или без него), подобную передаточной функции хорошо известного триггера Шмитта. Если сигнал на входе меньше некоторого порога, то на выходе получается сигнал одного уровня (обычно низкого), а если порог превышен, то сигнал на выходе становится другого уровня (обычно высокого).Если при спаде сигнала достигается другой порог, то сигнал на выходе также скачком меняется. Рису-

нок 7.4 показывает работу релейного блока с одинаковыми по абсолютной величине и очень малыми (eps) порогами при подаче синусоидального сигнала на вход.

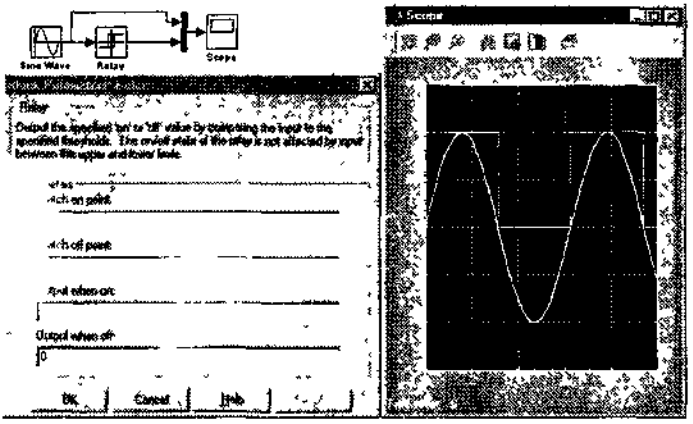


Рис. 7.4. Релейный блок и окно установки его параметров

В окне параметров блока можно задать уровни сигнала на выходе при включенном и выключенном состоянии, а также верхний и нижний пороговые уровни срабатывания. Их значения по умолчанию представлены на рис. 7.4.

Блок с ограничением скорости Rate Limiter

Блок с ограничением скорости Rate Limiter старается отслеживать за входным сигналом в условиях задания ограничений на скорость нарастания и спада сигнала (рис. 7.5).

Для вычисления скорости изменения сигнала используется соотношение:

$rate = [U_i - OUT_{i-1}]/[T_i - T_{i-1}]$

где i — текущий шаг моделирования, смысл остальных параметров очевиден. При работе блока вычисленное по этой формуле значение скорости изменения сигнала сравнивается с установленным в окне параметров значением параметра R (Rising slew rate). Если скорость изменения входного сигнала выше, чем заданная, то выходной сигнал «отрывается» от входного и меняется в соответствии с выражением:

$OUT_i = dT*R+OUT_{i-1}$,

где dT — приращение времени на текущем шаге модельного времени.

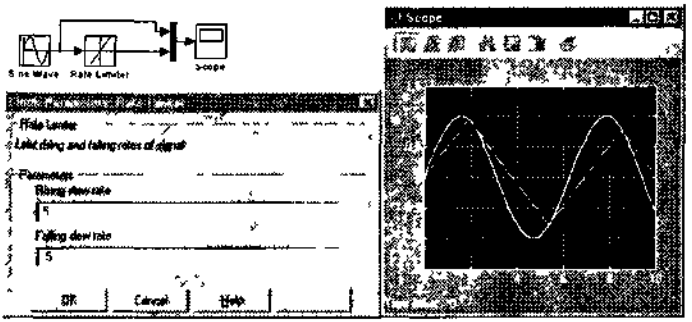


Рис. 7.5. Релейный блок и окно установки его параметров

Если вычисленная скорость меньше параметра F (Falling slew rate), то выходной сигнал меняется в соответствии с выражением:

$OUT = dT*F+OUT_{-1}$

Наконец, если вычисленная скорость находится в промежутке между значениями R и F, то входной сигнал повторяется выходным, то есть имеет место их равенство.

В окне параметров блока задаются скорости нарастания Rising slew rate и Falling slew rate По умолчанию заданы значения 0.5 и −0.5.

Блок следящего квантования Quantizer

Блок Quantizer служит для квантования меняющихся сигналов с одинаковым шагом по уровню (рис. 7.6).

Блок имеет единственный параметр — шаг по уровню (по умолчанию 0.5). Рисунок 7.6 показывает квантование синусоидального сигнала. Можно отметить, что при большом шаге квантования его трудно назвать идеальным — точного слежения за уровнем входного сигнала нет.

Блок фрикционных эффектов Coulombic and Viscous Friction

Блок фрикционных эффектов Coulombic and Viscous Friction служит для моделирования фрикционных эффектов (рис. 7.7). Передаточная функция блока указана в окне установки его параметров.

В качестве параметра блока задается список смещений при фрикционных эффектах и коэффициент передачи для приращений выходного сигнала.

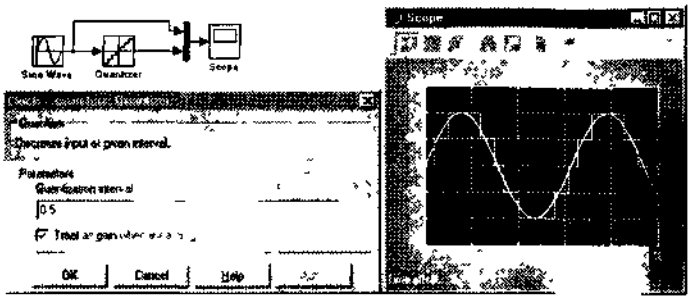


Рис. 7.6. Блок Quantizer и окно установки его параметров

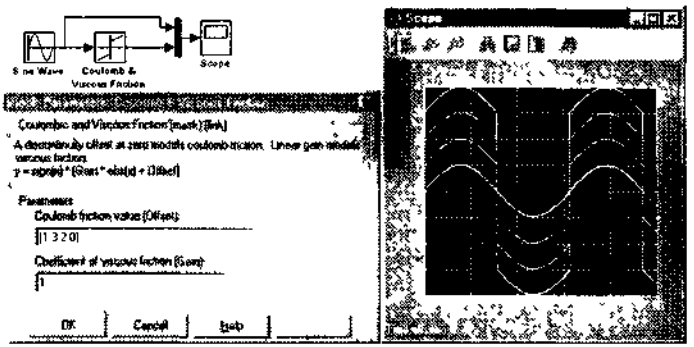


Рис. 7.7. Блок фрикционных эффектов и окно установки его параметров

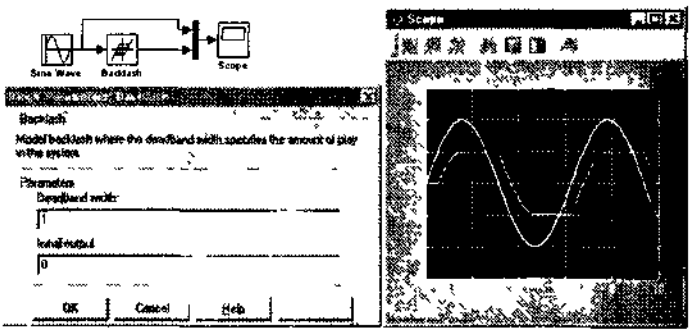


Рис. 7.8. Блок люфта и окно установки его параметров

Блок люфта Backlash

Блок Backlash имитирует эффект возникновения люфта (рис. 7 8). Этот эффект создает передаточную характеристику гистерезис-

ного типа, которая представляется графически в пиктограмме блока.

Блок имеет два параметра: ширина диапазона Deaband width и начальный уровень сигнала на выходе Initial output (по умолчанию 1 и 0). Уровень Initial output является также срединным значением входного сигнала, а Deaband width определяет ширину петли гистерезиса передаточной характеристики блока. Сигнал на входе будет равен заданному значению Initial output, пока при возрастании не достигнет значения U+(Deaband width)/2, после чего перестает меняться. При спаде сигнал перестает меняться, достигнув границы U–(Deaband width)/2.

Управляемый переключатель Switch

Управляемый переключатель Switch — это переключающее устройство с тремя входами: двумя крайними для сигналов данных и одним (средним) для сигналов управления. Если уровень сигнала управления превышает заданное значение, то на выход поступает сигнал с верхнего (первого) входа, иначе — с нижнего (второго) входа. Это поясняет пример применения блока, показанный на рис. 7.9.

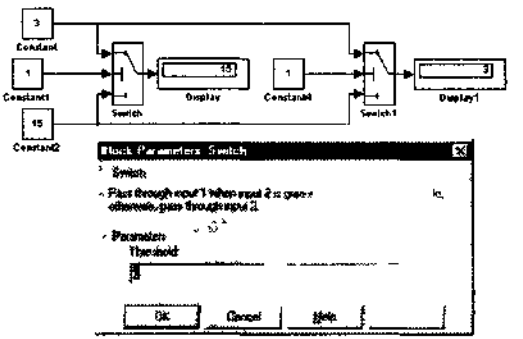


Рис. 7.9. Применение управляемого переключателя Switch

Единственным параметром ключа является порог управляющего сигнала Threshold (по умолчанию 0). Периодичность срабатывания ключа определяется эталонным временем. Ключ рассматривается как идеальное устройство, не имеющее остаточных параметров (таких, как переходное сопротивление в электрических ключах).

Блок многовходового переключателя Multiport Switch

Блок многовходового переключателя Multiport Switch в отличие от описанного выше переключателя позволяет задавать требуемое количество входов (портов) и подключает тот или иной вход к выходу в зависимости от уровня сигнала на управляющем входе (рис. 7.10).

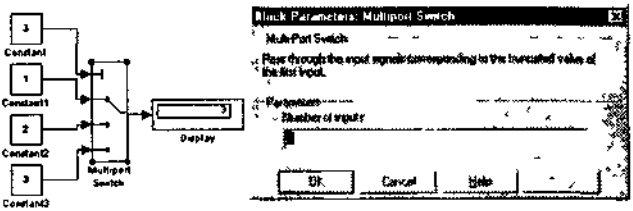


Рис. 7.10. Пример применения многовходового переключателя Multiport Switch

На рис. 7.10 на управляющий вход подана константа 3, поэтому к выходу подключен третий вход.

Если на управляющий вход подан сигнал, значение которого не совпадает с номером входа (порта), блок заключается в красную рамку, окрашивается и моделирование прекращается с выдачей сообщения об ошибке (рис. 7.11). Вообще говоря, подобный тип представления ошибок характерен для моделирования устройств и систем с помощью пакета Simulink.

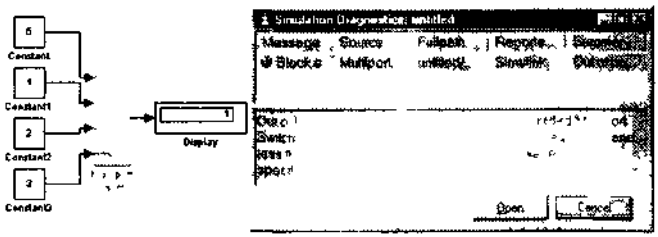


Рис. 7.11. Ошибка номера порта переключателя Multiport Switch

Единственным параметром этого блока является число его входов для данных Number of inputs. Многовходовой переключатель также рассматривается как идеальное устройство.

Дискретные блоки

Обзор дискретных блоков

Дискретные блоки служат для создания моделей дискретных устройств и систем двух типов: с дискретным временем и с дискретными состояниями. Эти блоки включают в себя устройства цифровой задержки, дискретно-временной интегратор, дискретный фильтр и т. п. На рис. 7.12 представлено окно библиотеки Discrete с этими устройствами.

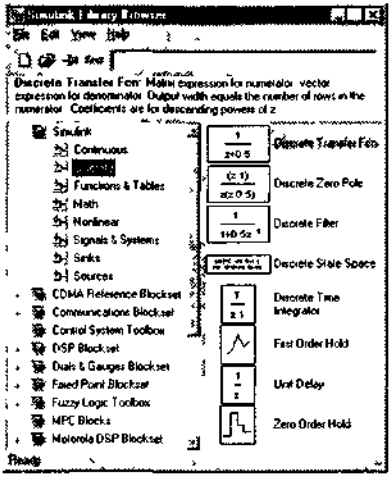


Рис. 7.12. Раздел библиотеки с дискретными блоками

Некоторые дискретные блоки имеют передаточные функции, напоминающие уже описанные для непрерывных блоков (см. главу 6). Однако основное отличие между ними заключается в том, что выходные сигналы дискретных блоков носят дискретный характер. Обычно при моделировании дискретных по времени устройств время моделирования имеет фиксированный шаг.

Блок дискретной единичной задержки Unit Delay

Блок Unit Delay обеспечивает задержку входного сигнала на один шаг эталонного времени. Это иллюстрирует рис. 7.13, на котором демонстрируется задержка перепада.

Окно параметров блока, также показанное на рис. 7.13, предусматривает установку начального значения для уровня выходного сигнала Initial condition и эталонного времени

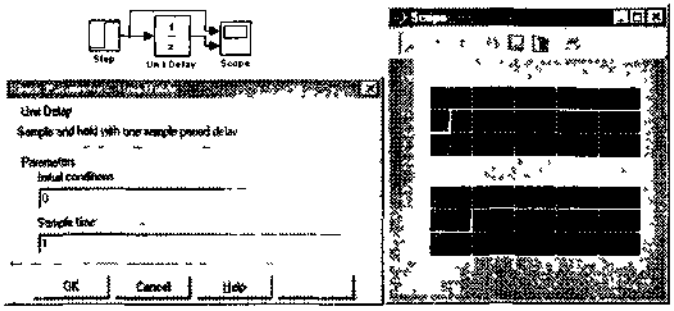


Рис. 7.13. Блок дискретной единичной задержки

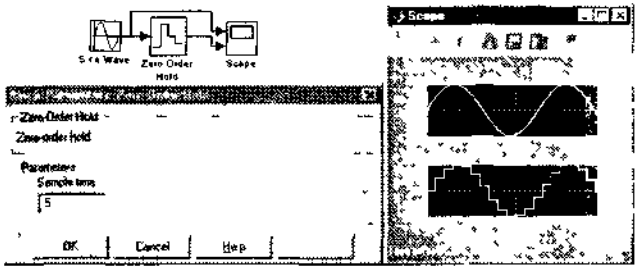


Рис. 7.14. Блок дискретного экстраполятора нулевого порядка

Блок экстраполятора нулевого порядка Zero-Order Hold

Новый блок экстраполятора нулевого порядка Zero-Order Hold задерживает выходной сигнал на заданный промежуток времени и оставляет неизменными значения выходного сигнала на каждом такте дискретизации (рис. 7.14).

Предусмотрена установка единственного параметра — эталонного времени.

Блок экстраполятора первого порядка First-Order Hold

Блок экстраполятора первого порядка First-Order Hold задерживает выходной сигнал на заданный промежуток времени и задает линей-

ное изменение выходного сигнала на каждом такте дискретизации, в соответствии с крутизной входного сигнала в данный момент времени. Пример задержки синусоидального сигнала этим блоком показан на рис. 7.15.

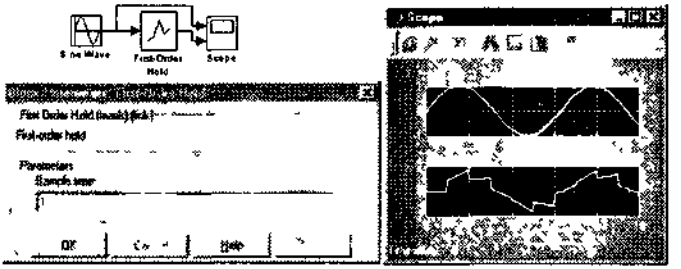


Рис. 7.15. Блок дискретного экстраполятора первого порядка

В окне параметров этого блока (рис. 7 15) предусмотрена установка единственного параметра — эталонного времени (по умолчанию 1). Обратите внимание, что оно может быть дробным числом.

Блок дискретного интегратора времени Discrete-Time Integrator

Блок Discrete-Time Integrator служит для дискретного интегрирования времени (рис. 7.16). Обычно он служит для управления логикой работы модели, например для остановки процесса моделирования по заданному значению интеграла времени.

Окно параметров дискретного интегратора представлено на рис. 7.16. Основными являются следующие параметры:

- Integration method — метод численного интегрирования;
- Sample time — эталонное время.

Метод интегрирования выбирается из раскрывающегося списка. Возможен выбор одного из трех методов:

- Forward Euler — прямой метод Эйлера;
- Backward Euler — обратный метод Эйлера;
- Trapeziodal — метод трапеций.

Расчетные соотношения для каждого из этих методов можно найти в справочной системе Simulink — они не приводятся, поскольку знания таких соотношений для практического моделирования, как правило,

не требуется (достаточно знать суть соответствующего метода). На рис. 7.16 надпись внутри графического изображения блока дана для методов Эйлера, для метода трапеций она иная. Назначение прочих параметров то же, что и для непрерывного интегратора (см. главу 6).

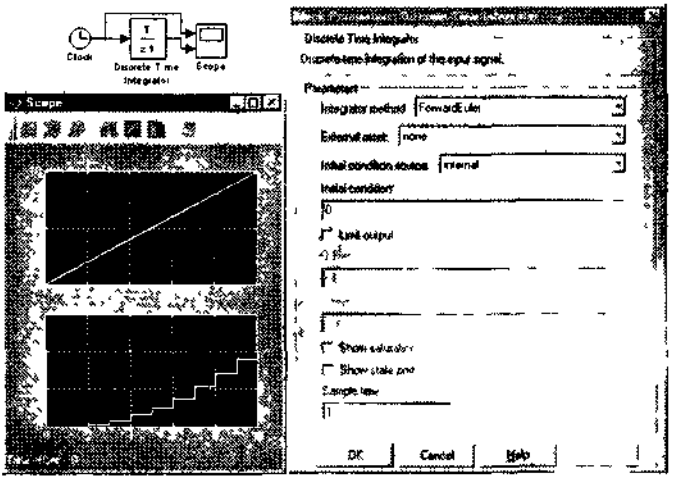


Рис. 7.16. Блок дискретного интегратора времени

Блок дискретного фильтра Discrete Filter

Блок Discrete Filter служит для создания дискретного фильтра, порядок и свойства которого задаются полиномом от частного 1/z в числителе передаточной характеристики (рис. 7.17).

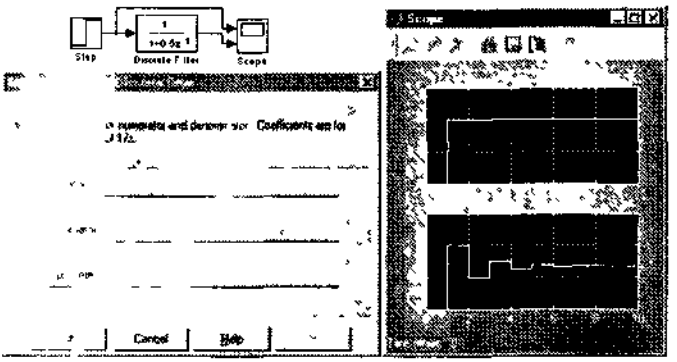


Рис. 7.17. Блок дискретного фильтра

Параметрами дискретного фильтра являются векторы, содержащие коэффициенты полиномов числителя и знаменателя, и эталонное время. Надпись в графическом представлении блока зависит от выбранного полинома.

Другие дискретные блоки

В состав библиотеки дискретных блоков входят также следующие блоки

- Discrete Transfer Fcn — блок дискретной передаточной функции;
- Discrete Zero Pole — блок задания дискретной области перехода,
- Discrete State Space — блок формирования дискретного пространства состояний

Их назначение и параметры аналогичны описанным ранее (см главу 6) для подобных по назначению непрерывных устройств

Глава 8. Библиотека Simulink Extras

- Обзор библиотеки Simulink Extras
- Дополнительные дискретные блоки Additional Discrete
- Дополнительные линейные блоки Additional Linear
- Блоки триггеров Flip Flops
- Раздел библиотеки Linearization
- Раздел библиотеки Transformations
- Раздел библиотеки Aerospace Block

Обзор библиотеки Simulink Extras

Библиотека Simulink Extras является дополнительной библиотекой пакета Simulink, но в отличие от ряда расширений этого пакета входит в состав его поставки. Эта библиотека содержит наборы блоков с более широкими функциями, чем рассмотренные в главах 5, 6 и 7 разделы основной библиотеки. Тем не менее это вовсе не означает, что применение этой библиотеки всегда предпочтительнее. Связано это с тем, что усложнение функций блоков, полезное при решении ряда специфических задач, оборачивается усложнением моделирования при решении большинства обычных задач.

В связи с этим область применения библиотеки Simulink Extras не очень широка, и мы рассмотрим блоки этой библиотеки не так подробно, как блоки основной библиотеки. Такой подход оправдан и тем, что многие дополнения в библиотеке Simulink Extras носят частный и исключительный характер. Именно в плане отличий от основной библиотеки и будет рассмотрена библиотека Simulink Extras.

Библиотека Simulink Extras представлена на рис. 8 1. Работа с этой библиотекой ничем не отличается от работы с основной библиотекой.

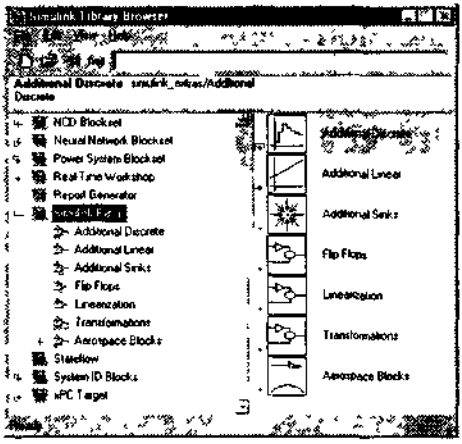


Рис. 8.1. Раздел библиотеки Simulink Extras

Как видно из рис. 8.1, библиотека Simulink Extras имеет следующие наборы блоков:

- Additional Discrete — дополнительные дискретные блоки;
- Additional Linear — дополнительные линейные блоки;
- Additional Sinks — дополнительные получатели сигналов (регистраторы),
- Flip-Flops — триггерные блоки;
- Linearization — линеаризирующие блоки;
- Transformations — блоки преобразования;
- Aerospace Blocks — блоки моделирования авиационных систем.

Дополнительные дискретные блоки Additional Discrete

Дополнительные дискретные блоки Additional Discrete представлены всего четырьмя блоками — по два варианта уже известных нам блоков Discrete Transfer Fcn и Discrete Zero Pole (рис. 8.2). Их единственным отличием от описанных ранее блоков является возможность инициализации входов и состояний.

В связи с этим ограничимся одним примером применения блока Discrete Transfer Fcn (with initial states) с инициализацией состояний (рис. 8.3).

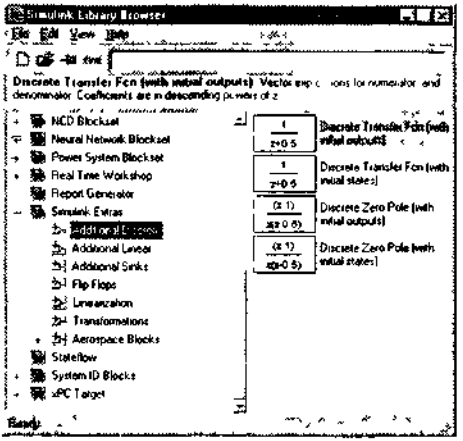


Рис. 8.2. Раздел библиотеки Simulink Extras с дискретными блоками

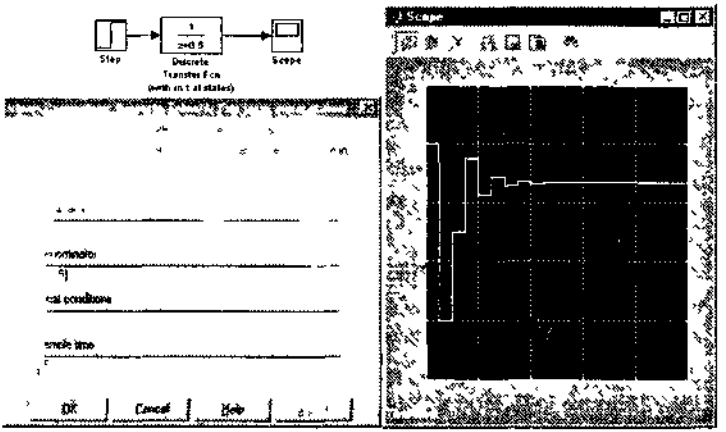


Рис. 8.3. Пример применения блока Discrete Transfer Fcn с инициализацией состояний

Дополнительные линейные блоки Additional Linear

Обзор дополнительных линейных блоков

Состав дополнительных линейных блоков раздела Additional Linear показан на рис. 8.4.

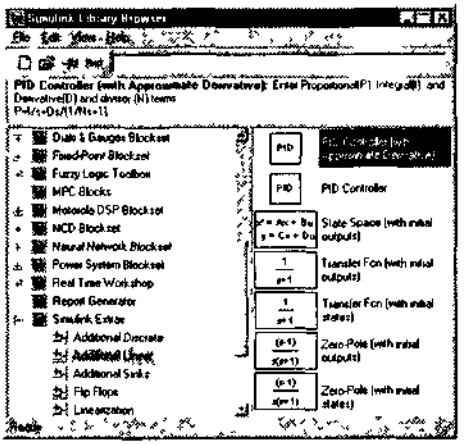


Рис. 8.4. Раздел библиотеки Simulink Extras с дополнительными линейными блоками

Блоки этого раздела можно разделить на две категории: PID-контроллеры и блоки типа State-Space, Transfer Fcn и Zero-Pole, дополненные возможностями инициализации выходных сигналов и состояний.

Блок PID-controller

PID-контроллер (PID controller) — это довольно универсальный блок, выходной сигнал которого задается операторным выражением:

```
OUT = P + I/s + Ds,
```

где P — входной сигнал, I — его интеграл и D — его производная. Параметр P (по умолчанию 1) фактически задает компоненту выходного сигнала, пропорциональную входному сигналу, параметр I задает пропорциональность интегралу входного сигнала и, наконец, параметр D задает пропорциональность производной входного сигнала. Варьируя параметры P, I и D, можно задавать различный вид выходного сигнала, в том числе в виде интеграла или производной от входного.

На рис. 8.5 дан пример применения PID-контроллера при входном сигнале в виде прямоугольных импульсов. Выходной сигнал задан как сумма входного сигнала и его производной. Установка параметра I = 0 исключает интегральную компоненту.

Нетрудно заметить, что операция дифференцирования выполняется данным PID-контроллером довольно грубо — на фронтах импульсов вместо пиков ничтожной длительности и бесконечной амплитуды видны пики вполне конечной амплитуды и длительности.

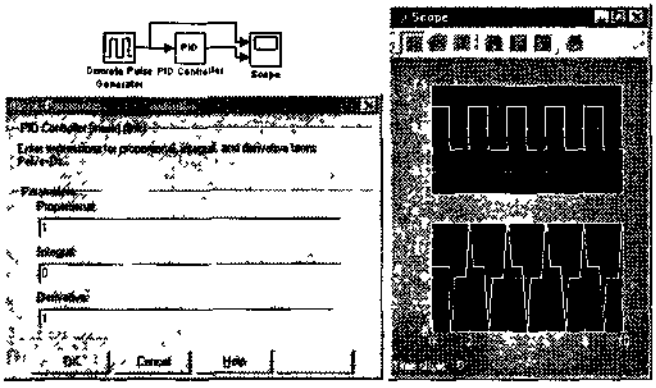


Рис. 8.5. Пример применения PID-контроллера

Блок PID-controller с улучшенной операцией дифференцирования

Другой блок — PID controller with Approximate Derivative (PID-контроллер с улучшенным вычислением производной) вычисляет выходной сигнал как

```
OUT = 3 + I/s + Ds/(1/Ns+1)
```

За счет применения дополнительного параметра N улучшается вычисление производной.

Рисунок 8.6 показывает работу этого блока при отключенном вычислении производной — выходной сигнал представлен суммой входного сигнала и его интеграла (см. установки в окне параметров этого блока на рис. 8.6).

Другой пример (рис. 8.7) показывает применение улучшенного PID-контроллера при входном сигнале в виде прямоугольных импульсов. Здесь отключена операция интегрирования, но включена операция дифференцирования. Это видно из установок в окне параметров улучшенного PID-контроллера.

Нетрудно заметить, что в данном случае дифференцирование входных прямоугольных импульсов хотя тоже не идеально, но намного лучше, чем в примере, представленном на рис. 8.5.

Остальные блоки раздела Additional Linear библиотеки Simulink Extras повторяют по функциям уже описанные блоки раздела Continuous основной библиотеки и отличаются только возможностью инициализации выходных сигналов или состояний.

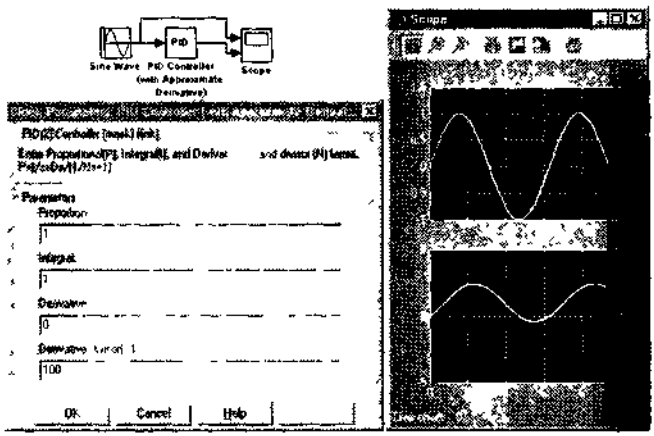


Рис. 8.6. Пример применения улучшенного PID-контроллера при обработке синусоидального сигнала

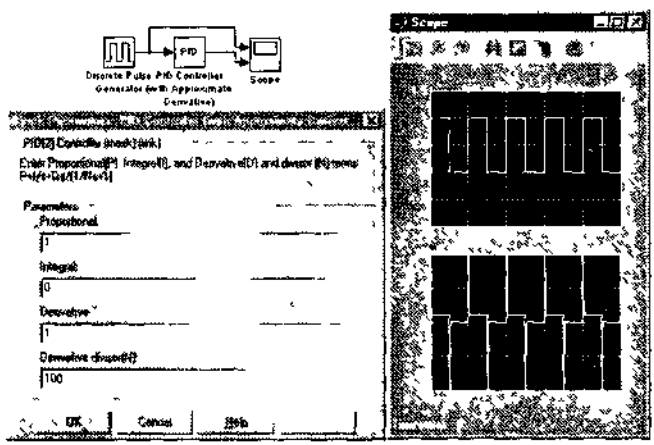


Рис. 8.7. Пример применения улучшенного PID-контроллера при обработке сигнала в виде прямоугольных импульсов

Дополнительные блоки Additional Sinks

Раздел дополнительных блоков Additional Sinks содержит ряд новых виртуальных регистраторов (рис. 8.8).

В него входят следующие блоки:

- Auto Correlator — автокоррелятор (используется с пакетом Signal Processing Toolbox);

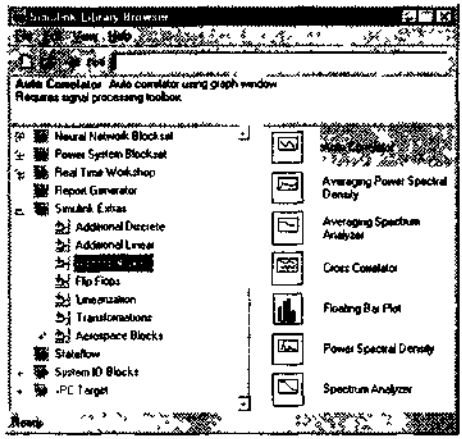


Рис. 8.8. Раздел дополнительных блоков Additional Sinks

- Averaging Power Spectral Density — анализатор спектральной плотности мощности с усреднением;
- Averaging Spectrum Analyzer — спектральный анализатор с усреднением;
- Cross-Correlator — кросс-коррелятор;
- Floating Point Bar — утилита построения гистограммы;
- Power Spectral Density — анализатор спектральной плотности мощности;
- Spectrum Analyser — анализатор спектра.

Нетрудно заметить, что данные блоки относятся к двум важным разделам моделирования — статистическому анализу и анализу спектров сигналов (по уровню и по мощности).

Блоки спектрального анализа

Рассмотрим как наиболее характерный блок Averaging Power Spectral Density. Он служит для наглядного представления формы сигналов и их энергетического спектра (рис. 8.9). Спектр (в данном случае прямоугольных импульсов) представляется амплитудно-частотной характеристикой мощности с усреднением и фазо-частотной характеристикой.

Блок имеет следующие параметры:

- Length of buffer — размер буфера;

- Number of point for fft — число точек для быстрого преобразования Фурье;
- Plot after how many points — построение графика после задания числа точек;
- Sample time — эталонное время.

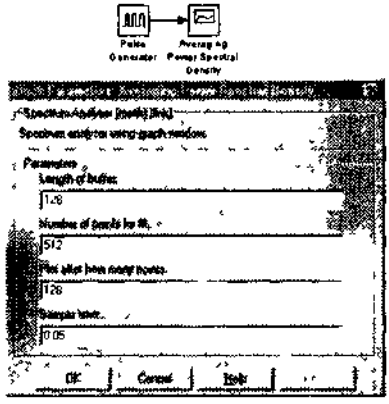


Рис. 8.9. Пример применения блока Averaging Power Spectral Density

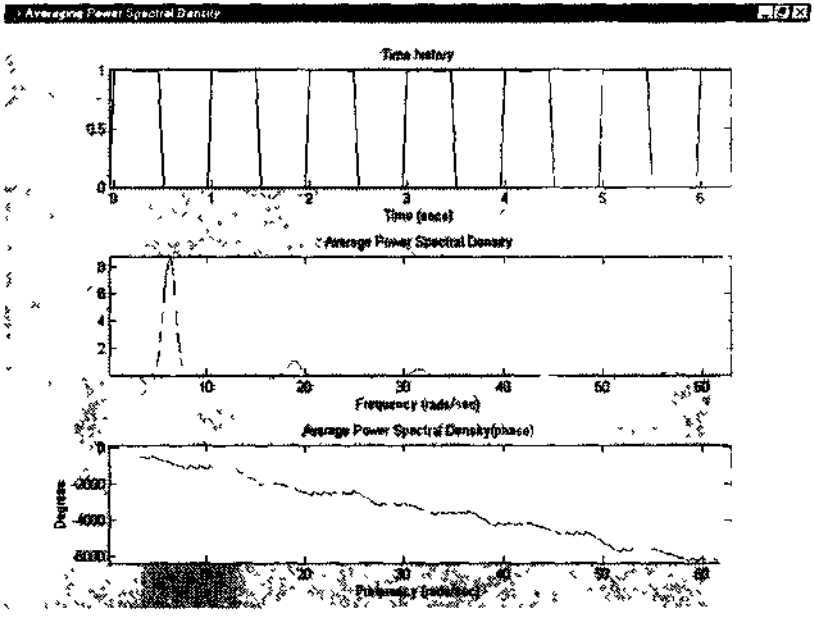


Рис. 8.10. Графики, создаваемые блоком Averaging Power Spectral Density

На рис 8.10 представлено полностью открытое окно анализатора спектра. Оно содержит три графика:

- график временной зависимости входного сигнала;
- график амплитудно-частотной характеристики мощности входного сигнала;
- график зависимости фазы от частоты.

Масштабы графиков линейные. Спектр вычисляется приближенно на основе быстрого преобразования Фурье (БПФ). Нетрудно заметить, что спектрограмма гораздо больше напоминает кривые, созданные реальными анализаторами спектра, чем получаемые теоретически. В частности, линии первой и высших (в данном случае нечетных) гармоник представлены колоколообразными кривыми, а не четкими вертикальными линиями. Как достоинство спектрограмм можно отметить широкий диапазон представления углов без характерных разрывов фазы, возникающих при ограничении изменения фазы, например в диапазоне углов от 0 до 360 градусов (от 0 до 2π радиан).

Остальные анализаторы спектра используются аналогично и имеют такое же окно представления результатов спектрального анализа. Их различия вытекают из наименования блоков (см. выше).

Блок кросс-коррелятора Cross-Correlator

Блок Cross-Correlator служит для построения графиков поданных на его вход сигналов и текущего коэффициента корреляции, отражающего степень взаимосвязи изменений сигналов. Рисунок 8.11 показывает применение этого блока в случае, когда на входе действуют сигналы разного вида — синусоида и пилообразный импульс — с одинаковыми частотами.

Параметры этого блока имеют то же назначение, что и одноименные параметры блока Averaging Power Spectral Density.

Блоки триггеров Flip Flops

Обзор раздела библиотеки Flip Flops

Раздел библиотеки Simulink Extras Flip Flops содержит генератор тактовых прямоугольных импульсов и 4 блока триггеров (рис. 8.12).

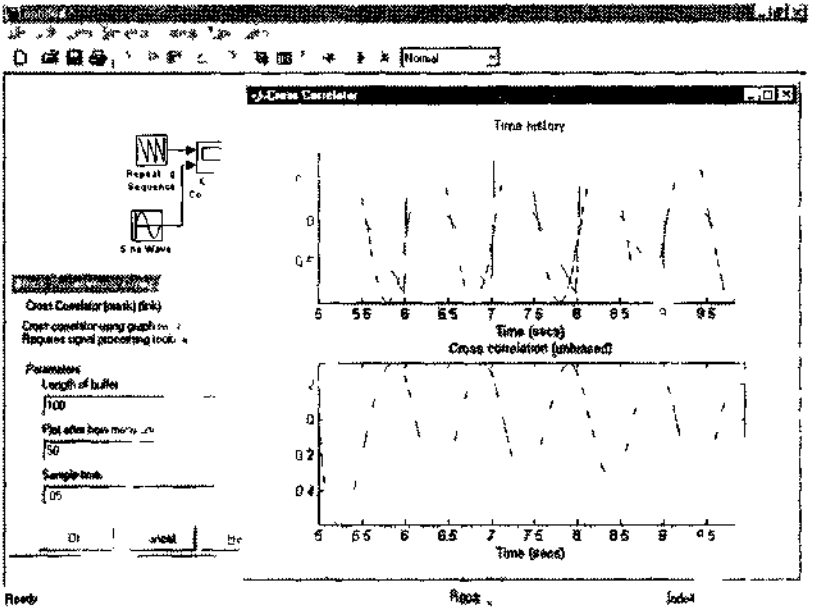


Рис. 8.11 Пример применения блока кросс-коррелятора

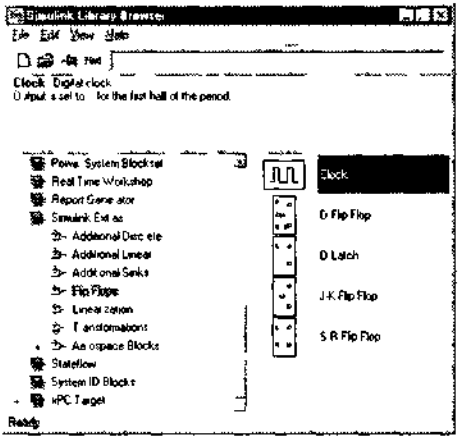


Рис. 8.12. Раздел библиотеки Flip Flops

Генератор тактовых импульсов Clock

Генератор Clock является самым простым из источников прямоугольных импульсов. Он создает сигнал в виде импульсов единичной

амплитуды при скважности, равной 2 (напоминаем, что скважность есть отношение периода импульсов к их длительности) Пример применения блока Clock представлен на рис 8 13

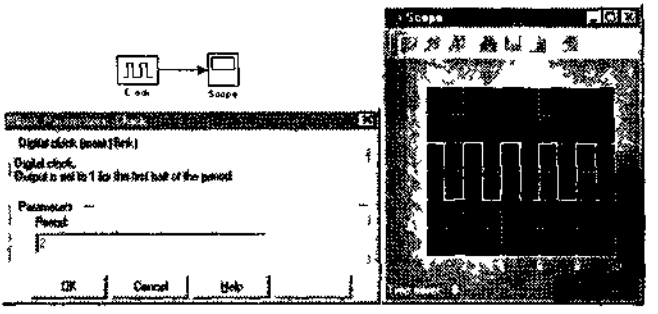


Рис. 8.13. Пример применения блока Clock

Единственным параметром этого источника импульсов является период (в тактах периода эталонного времени) Этот источник обычно используется для тактирования работы триггеров

Триггерные блоки

Раздел библиотеки Flip Flops содержит 4 блока триггерных устройств, реализующих следующие типы стандартных триггерных устройств D Flip Flop, D Latch, J-K Flip-Flops и S-R Flip-Flops Эти блоки и соответствующие им окна параметров показаны на рис 8 14 D-триггеры не имеют параметров, а для JK- и RS-триггеров задается единственный параметр — уровень инициализации

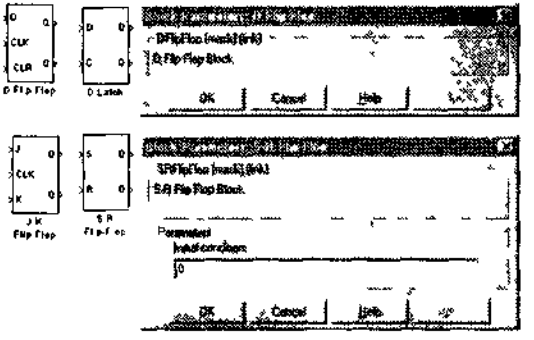


Рис. 8.14. Триггерные блоки и окна установки их параметров

Пример построения широтно-импульсного модулятора

Триггеры часто используются для построения импульсных устройств. На рис. 8.15 дан пример построения модели широтно-импульсного модулятора (ШИМ) на основе RS-триггера

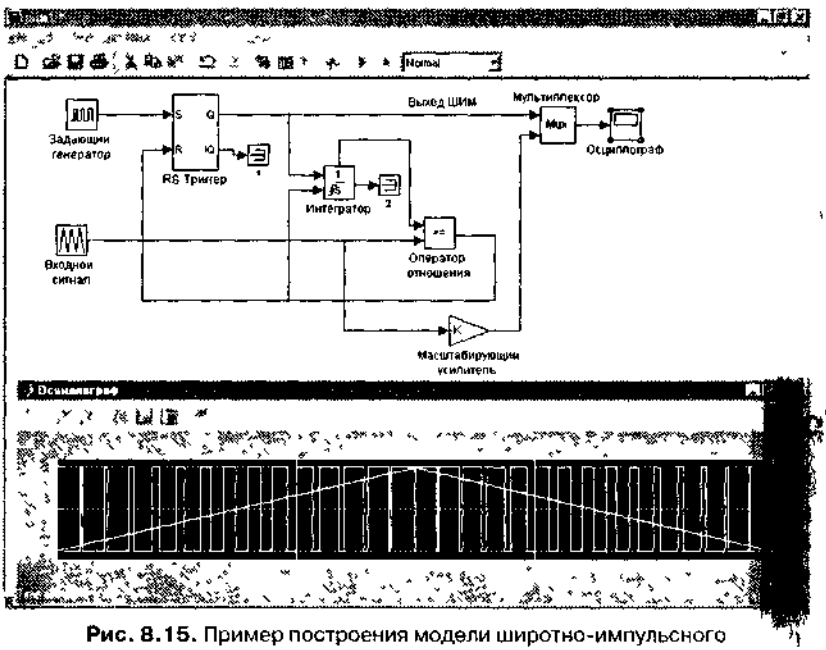


Рис. 8.15. Пример построения модели широтно-импульсного модулятора

8

Работа модулятора основана на сравнении быстро изменяющегося пилообразного сигнала, созданного релаксационной замкнутой системой на базе RS-триггера и интегратора, с медленно изменяющимся треугольным входным сигналом. Это меняет скважность импульсов, создаваемых на одном из выходов триггера. Обратите внимание на применение в этой модели заглушек (терминаторов) на неиспользуемых выходах триггера и интегратора. Для интегратора следует установить параметр External reset равным rising (сброс по положительному перепаду управляющего сигнала), кроме того, должен быть установлен флажок Show state port, задающий появление дополнительного выхода — выхода состояния.

Эта модель является примером моделирования реальной электронной схемы с помощью стандартных средств пакета Simulink. Подобные модуляторы в настоящее время выпускаются в виде интегральных микросхем. Они используются в измерительных устройствах и в импульсных преобразователях электрической энергии.

Раздел Linearization

Раздел Linearization представлен на рис. 8.16.

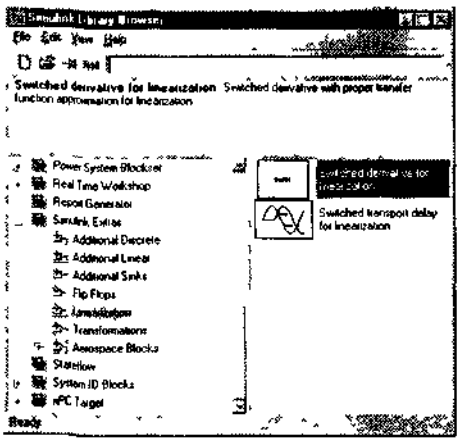


Рис. 8.16. Раздел библиотеки Linearization

Как нетрудно заметить, этот раздел содержит всего два блока — дифференцирующий блок и блок заданной временной задержки.

Работу дифференцирующего блока поясняет рис. 8.17. Здесь показан случай дифференцирования прямоугольных импульсов от источника Clock, описанного выше. Качество дифференцирования весьма низкое. Это связано с тем, что установлен параметр Switch value = 0, что отключает линеаризацию процесса дифференцирования.

Достаточно установить указанный параметр равным 1, и качество дифференцирования намного возрастает — см. пример на рис. 8.18.

Помимо указанного параметра в этом блоке задается еще один параметр — константа дифференцирования (она входит в знаменатель операторной передаточной функции блока).

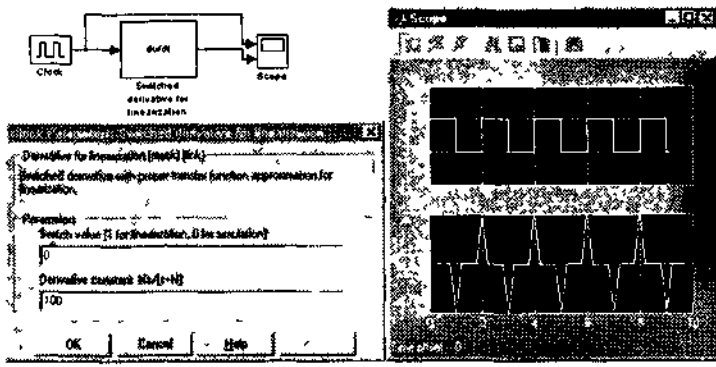


Рис. 8.17. Пример работы блока дифференцирования

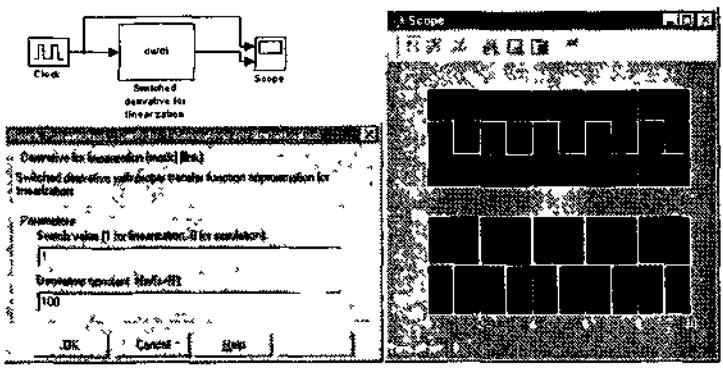


Рис. 8.18. Пример дифференцирования с повышенным качеством

Блок заданной временной задержки

Блок заданной временной задержки Switched transport delay for linearization представляет собой линейное устройство временной задержки. Длительность задержки задается параметром этого блока Time delay. Применение блока для задержки синусоидального сигнала иллюстрирует рис. 8.19.

В окне параметров временной задержки помимо параметра Time delay устанавливается начальное значение входного сигнала Initial input, начальная емкость буфера Initial buffer size и порядок Паде-линеаризации.

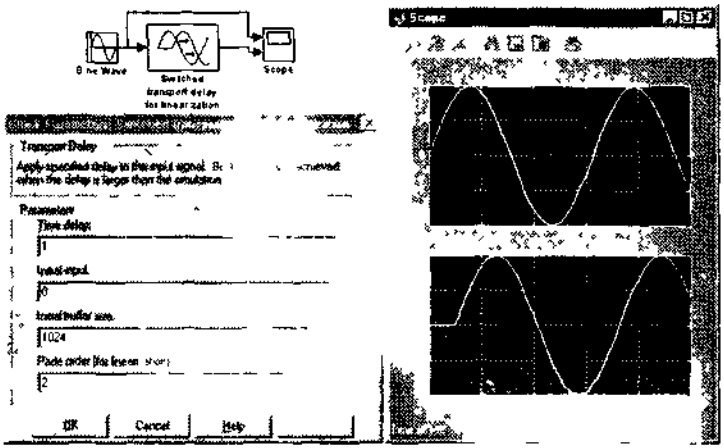


Рис. 8.19. Пример применения временной задержки

Раздел преобразований Transformations

Обзор раздела преобразований

Раздел преобразований Transformations содержит 8 блоков для осуществления типичных преобразований — это преобразование температуры, углов и систем координат (рис. 8.20).

Соответствующие формулы преобразования появляются в информационном поле окна этого раздела, а также в окнах параметров блоков. В связи с этим расчетные формулы преобразований не приводятся.

Блок преобразования температуры Celsius to Fahrenheit

Работа блока Celsius to Fahrenheit, который преобразует температуру, выраженную в градусах Цельсия, в температуру, выраженную в градусах Фаренгейта, демонстрирует рис. 8.21 (сверху).

Блок преобразования температуры Fahrenheit to Celsius

Работу блока Fahrenheit to Celsius, переводящего температуру из градусов Фаренгейта в градусы Цельсия, показывает рис. 8.21 (снизу).

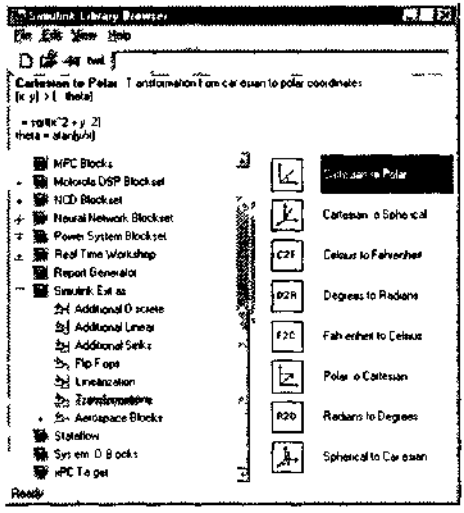


Рис. 8.20. Раздел библиотеки с блоками преобразований

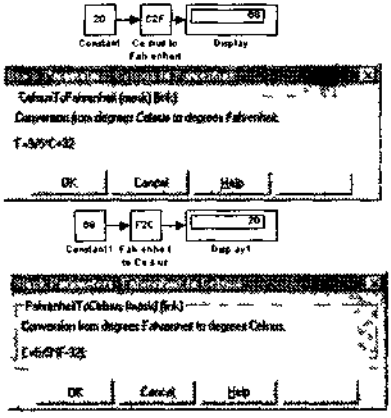


Рис. 8.21. Примеры преобразования температуры

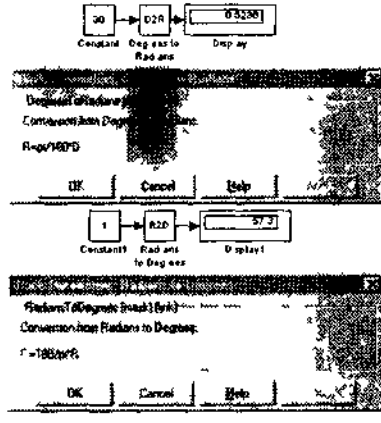


Рис. 8.22. Примеры преобразования углов

Блок преобразования углов Degress to Radians

Блок Degress to Radians служит для преобразования углов, выраженных в градусах, в углы, выраженные в радианах. Работу блока поясняет рис. 8.22 (сверху)

Блок преобразования углов Radians to Degress

Блок Radians to Degress служит для преобразования углов, выраженных в радианах, в углы, выраженные в градусах Работу блока поясняет рис 8 22 (снизу)

Блок преобразования координат Cartesian to Polar

Блок Cartesian to Polar служит для преобразования прямоугольных координат точки на плоскости в полярные координаты (рис 8 23)

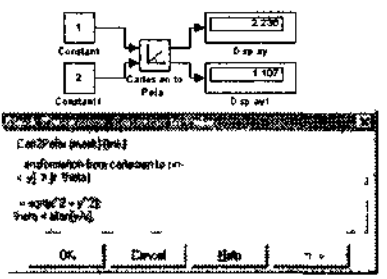


Рис. 8.23. Пример преобразования прямоугольных координат точки на плоскости в полярные координаты

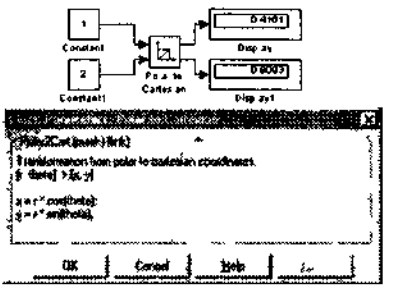


Рис. 8.24. Пример преобразования полярных координат точки на плоскости в прямоугольные координаты

Блок преобразования координат Polar to Cartesian

Блок Polar to Cartesian служит для преобразования полярных координат точки в прямоугольные координаты на плоскости (рис 8 24)

Блок преобразования 3D-координат Cartesian to Spherical

Блок Cartesian to Spherical служит для преобразования прямоугольных координат точки в пространстве в сферические координаты (рис 8 25)

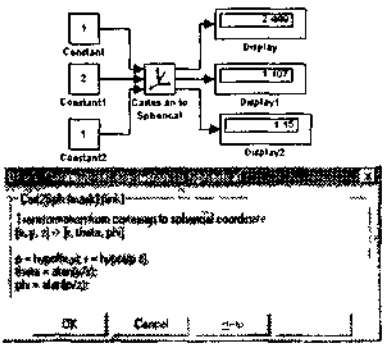


Рис. 8.25. Пример преобразования прямоугольных координат точки в пространстве в сферические координаты

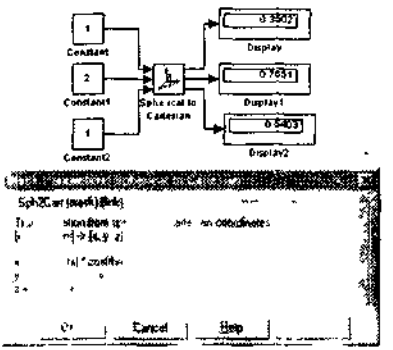


Рис. 8.26. Пример преобразования сферических координат точки в пространстве в прямоугольные координаты

Блок преобразования 3D-координат Spherical to Cartesian

Блок Spherical to Cartesian служит для преобразования сферических координат точки пространства в прямоугольные координаты (рис. 8.26).

Раздел Aerospace Block

Обзор раздела Aerospace Block

Раздел Aerospace Block представлен на рис. 8 27. Этот довольно обширный раздел, имеющий шесть подразделов. Каждый подраздел открывается активизацией знака + у нижнего левого угла соответствующей пиктограммы.

Ввиду узкого, хотя и безусловно важного, значения блоков этого раздела мы не будем рассматривать их. Читатель, занимающийся проектированием летательных аппаратов, может самостоятельно ознакомиться с этим разделом библиотеки. Мы ограничимся лишь парой наглядных примеров применения блоков данного раздела.

Задание положения летательного аппарата в пространстве

Как известно, многие параметры летательных аппаратов (например, подъемная сила) зависят от их положения в пространстве На рис. 8.28

представлена модель летательного аппарата — «голубка» — в пространстве.

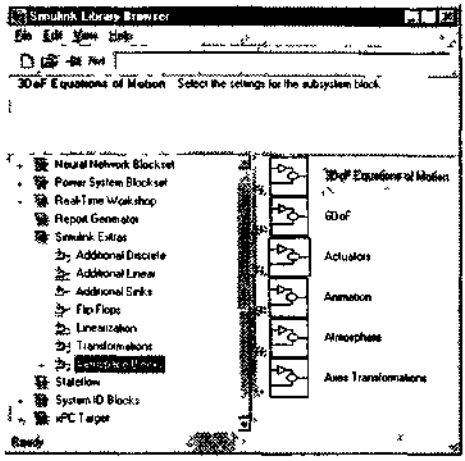


Рис. 8.27. Раздел Aerospace Block

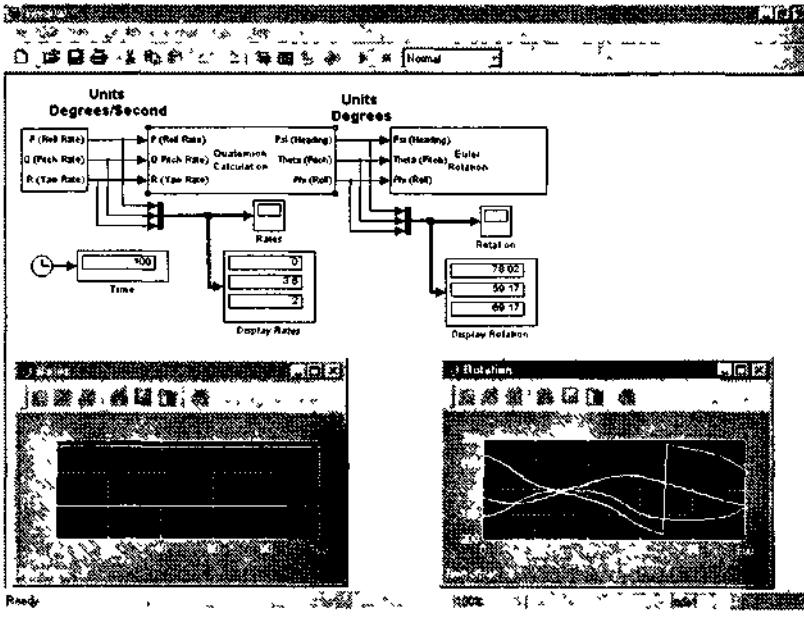


Рис. 8.28. Модель летательного аппарата в пространстве

Эта модель позволяет с помощью окна, имитирующего работу тренажера, устанавливать любое положение аппарата в пространстве и оценивать его параметры (рис. 8.29).

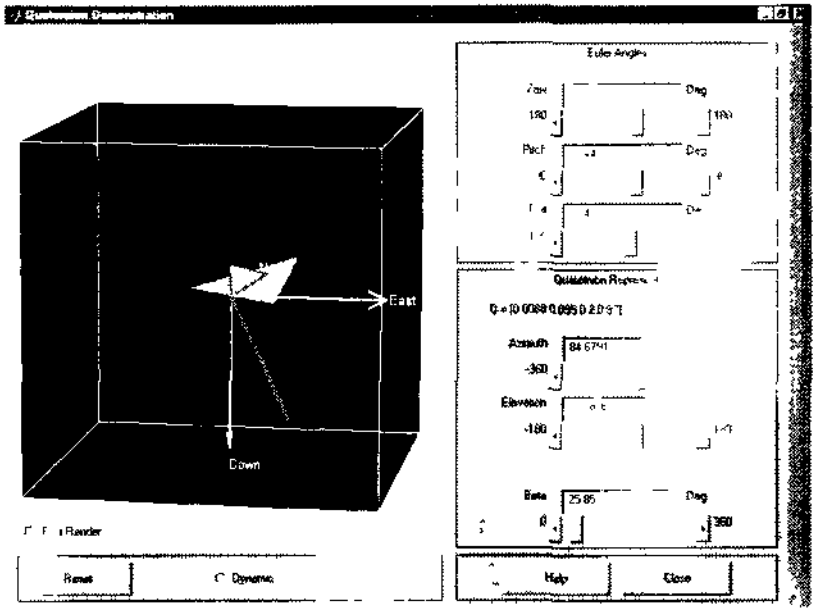


Рис. 8.29. Виртуальный тренажер для модели летательного аппарата

Здесь важно отметить применение простых средств анимации — изменение параметров аппарата с помощью виртуальных ползунков тут же сопровождается изменением положения аппарата в пространстве.

Система автоматического управления летательным аппаратом f14

Классическим стал пример моделирования системы управления летательным аппаратом f14 (рис. 8.30).

Модель имитирует (с помощью источника прямоугольных импульсов и константы) случай, когда пилот резко дергает штурвал. Система отрабатывает эти резкие изменения и наглядно (с помощью осциллограмм) отражает поведение летательного аппарата.

Данная модель является наглядной иллюстрацией применения подсистем (или субмоделей). Наиболее важные ее блоки представлены

подсистемами. Одна из таких подсистем (подсистема контроллера летательного аппарата) представлена на рис. 8 31.

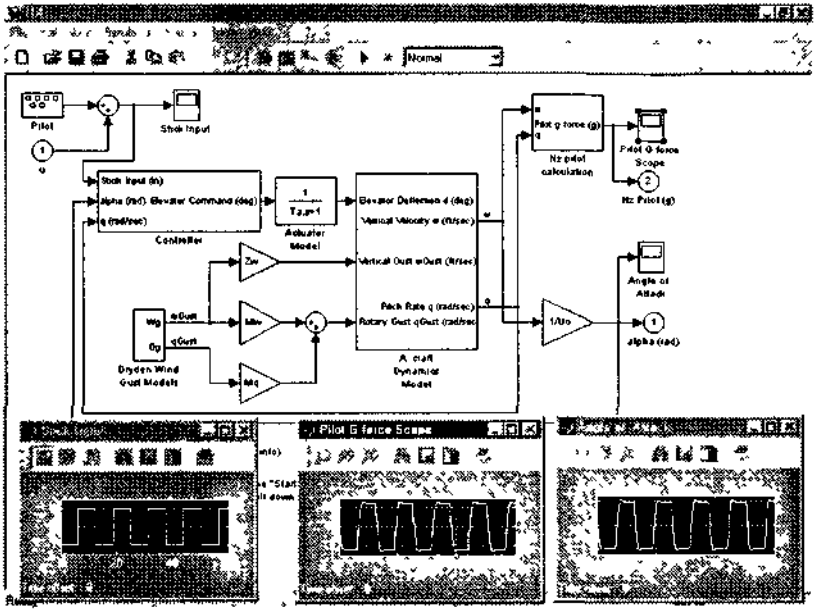


Рис. 8.30. Модель системы автоматического управления летательным аппаратом f14

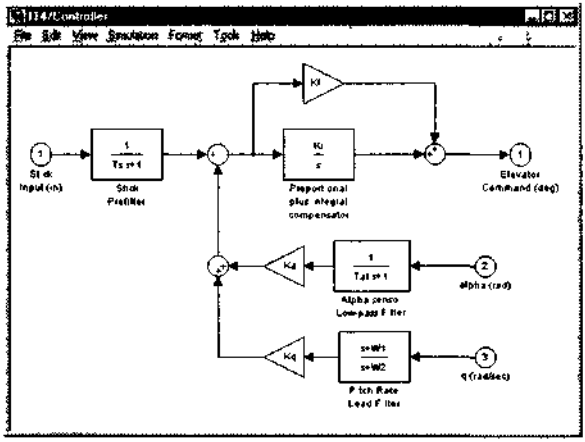


Рис. 8.31. Подсистема f14/Controller

Другая важная подсистема — подсистема динамики аппарата — представлена на рис. 8 32.

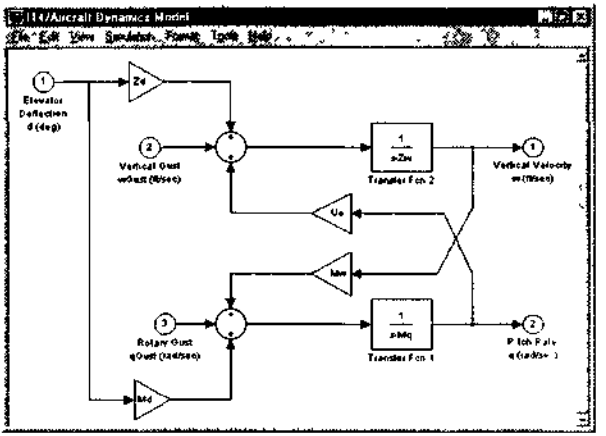


Рис. 8.32. Подсистема f14/Aircrast Dynamics Model

Благодаря применению подсистем основная модель (система) существенно упрощается, и из нее удаляются второстепенные детали. Простота вызова подсистем (двойной щелчок мыши при курсоре, указывающем на подсистему) позволяет в любой момент вызвать окно подсистемы для знакомства с ней и внесения изменений.

По аналогии с этим примером читатель может опробовать в работе и другие блоки данного раздела.

Глава 9. Создание подсистем

- Общие сведения о подсистемах
- Создание подсистемы из части основной модели
- Построение подсистем на основе блока SubSystem
- Управляемые подсистемы

Общие сведения о подсистемах

Как уже ясно, хотя бы из последнего примера главы 8 и примеров главы 3, пакет Simulink обеспечивает создание моделей, внутри которых располагаются подсистемы (субмодели). Внутри подсистем первого уровня, в свою очередь, могут располагаться подсистемы второго уровня и т. д. Это напоминает ситуацию, когда сложная система набирается из отдельных систем — модулей, каждый из которых, в свою очередь, является системой или устройством.

Такой принцип конструирования сложных моделей дает ряд важных достоинств:

- имеется возможность разбивки решаемой задачи на ряд более мелких задач, решаемых подсистемами;
- каждая подсистема может отлаживаться отдельно и использоваться в полной системе уже после отладки;
- существенно упрощается вид основной модели за счет исключения из нее второстепенных блоков;
- облегчается модификация полной модели за счет модификации ее более простых подсистем.

Создание подсистемы из части основной модели

Постановка задачи о выделении подсистемы

Simulink дает возможность выделить в любой модели некоторый блок и тут же превратить его в подсистему. Допустим, что мы решили создать блок, который преобразует входной сигнал в пять сигналов:

- сигнал, имитирующий люфт;
- квантованный сигнал;
- сигнал, характерный для реле;
- сигнал с ограничением;
- исходный сигнал.

Мы можем составить модель такого устройства, взяв соответствующие блоки, объединив их входы и используя мультиплексор Mux на выходе. Получим модель, представленную на рис. 9 1.

Запустив модель, можно наблюдать ее работу, что иллюстрируют осциллограммы виртуального осциллографа, подключенного к выходу мультиплексора.

В данном простом случае постановка задачи о выделении подсистемы заключается в оценке того, какие блоки мы планируем выделить в подсистему. Очевидно, что это будут нелинейные блоки, преобразующие сигнал, и мультиплексор. Для более сложных моделей решить, какие блоки выделяются в подсистемы, не так просто. Однако это уже задача пользователя, моделирующего конкретную систему или устройство.

Выделение блоков для подсистемы

Выделение блоков в подсистему выполняется довольно просто. Сначала надо установить курсор мыши около нужной группы блоков. Нажав левую кнопку и начав передвигать мышь, можно увидеть, что ее передвижение создает на экране прямоугольную рамку из черных точечных линий (рис. 9.1).

Отпустив левую кнопку мыши, можно наблюдать, как попавшие в прямоугольник блоки выделяются (рис. 9.2).

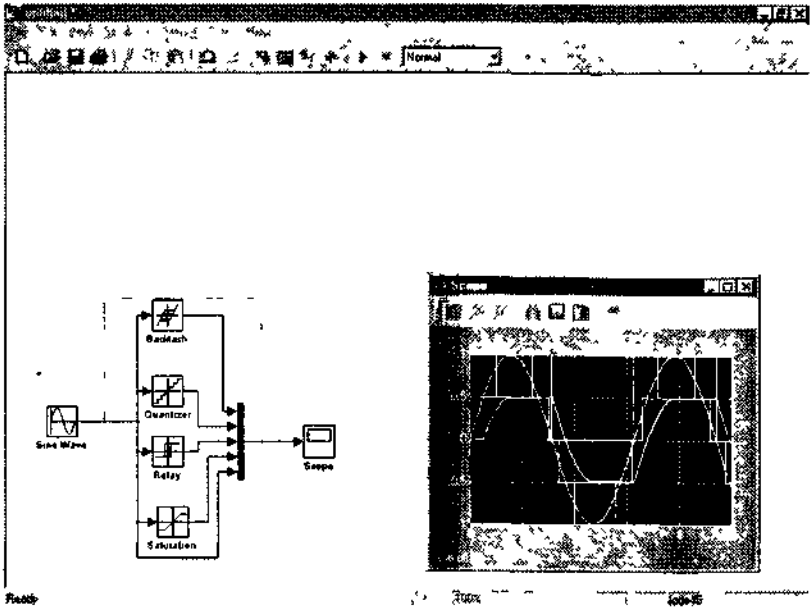


Рис. 9.1. Исходная модель устройства

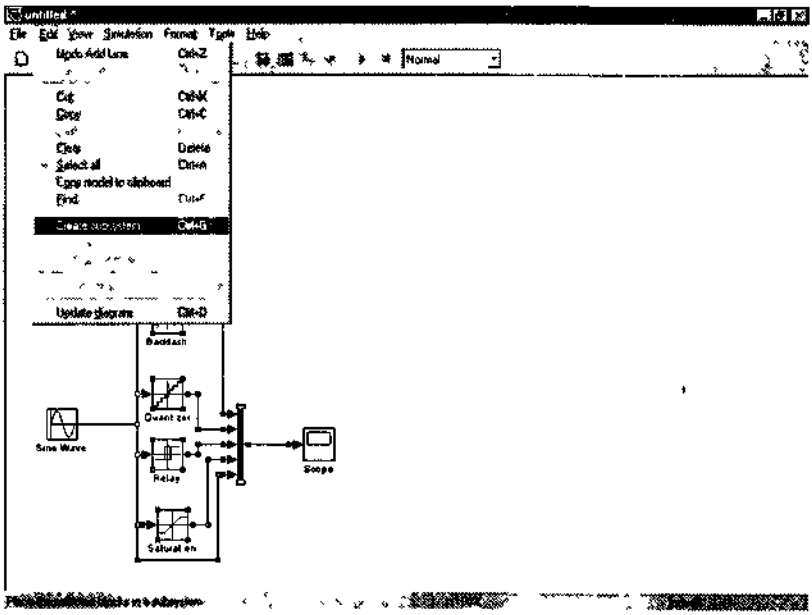


Рис. 9.2. Подготовка к созданию подсистемы

Теперь надо выбрать пункт меню Edit ▸ Create Subsystem (на рис. 9.2 меню Edit раскрыто)

Создание подсистемы из выделенных блоков

После выполнения команды Create Subsystem на месте выделенных блоков появится блок подсистемы. Обратите внимание на то, что для этой операции недоступна команда Undo (отмена последней операции). Поэтому перед выделением части модели в подсистему рекомендуется сохранить исходную модель под каким-либо новым именем с помощью команды Save As... меню File окна модели Simulink.

На рис. 9.3 показана полученная новая модель с подсистемой после редактирования модели переноса ее мышью в левый верхний угол окна и замены подписей блоков.

ВНИМАНИЕ

Обратите внимание на то, что эти надписи сделаны на русском языке Это приятная возможность пакета Simulink, но злоупотреблять ею не стоит, поскольку большинство терминов (в том числе названия блоков) имеет международный характер и выполнено на английском языке Но, пожалуй, главное — это то, что введение русскоязычных надписей способно вызвать серьезные сбои в работе модели Например, наблюдались такие «фокусы», как нарушение целостности записи имени после записи модели на диск и последующего ее считывания, а также восприятие части надписи как несуществующего параметра модели с выводом сообщения об ошибке

Вызов и просмотр подсистемы

Чтобы вызвать подсистему для просмотра или модификации, достаточно навести на нее курсор мыши и дважды щелкнуть левой кнопкой. Появится окно созданной подсистемы, показанное на рис. 9.3 внутри окна модели.

Назначение портов ввода и вывода в подсистемах

А теперь обратите внимание на главное — блок-схему полученной подсистемы. Нетрудно заметить, что состав блоков и соединений в подсистеме остался тем же, что и в исходной модели. Основное отличие проявляется в том, что в подсистеме автоматически появились новые блоки — порт входа In1 и порт выхода Out1. Порты изображаются овалом с номером внутри и подписями.

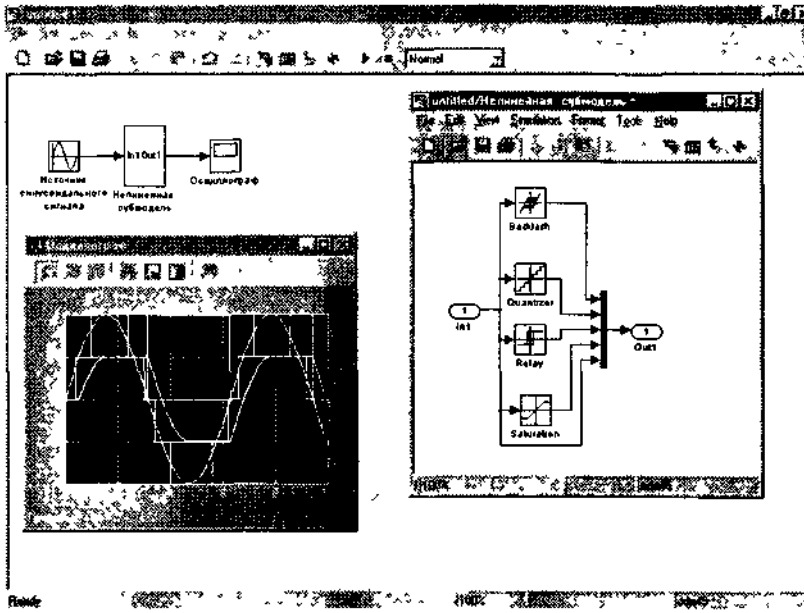


Рис. 9.3. Модель с подсистемой

Благодаря этим портам подсистема включается в состав основной модели. Если выделенный под подсистему блок содержит несколько входов и выходов, то в подсистеме появится несколько портов ввода и портов вывода. Они будут обозначены как `In1`, `In2`, `In3`, ... и `Out1`, `Out2`, `Out3`, ...

Порты входа служат для приема данных подсистемами, а порты выхода — для передачи выходных данных в основную модель (или в подсистему более высокого уровня). Заметим, что порты входа и выхода могут переименовываться пользователем при условии, что их имена будут уникальными.

Использование браузера моделей для работы с подсистемами

Окно подсистем полностью аналогично окну основной системы. Поэтому при работе с подсистемами возможно использование всех средств, которые имеются в этом окне. Например, можно запустить браузер модели, нажав соответствующую кнопку в панели инструментов субмодели (рис. 9.4).

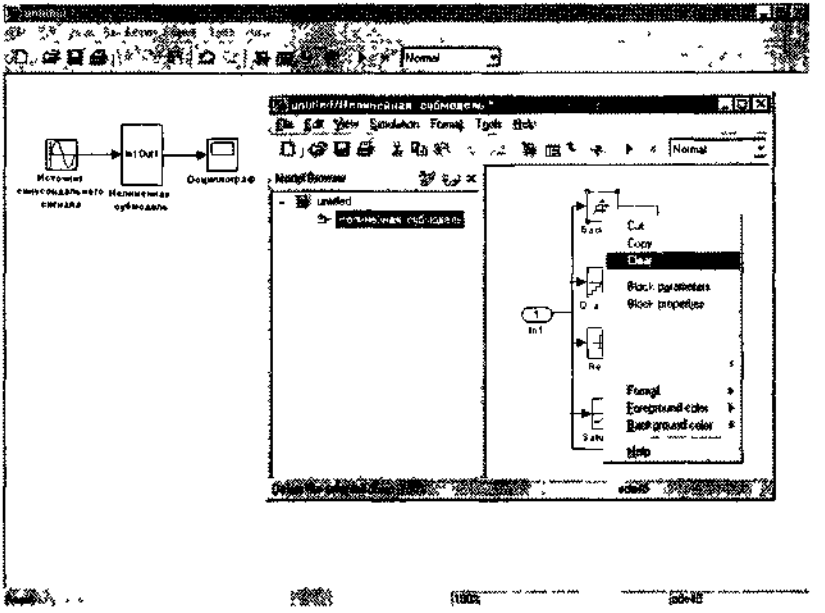


Рис. 9.4. Пример работы с браузером модели

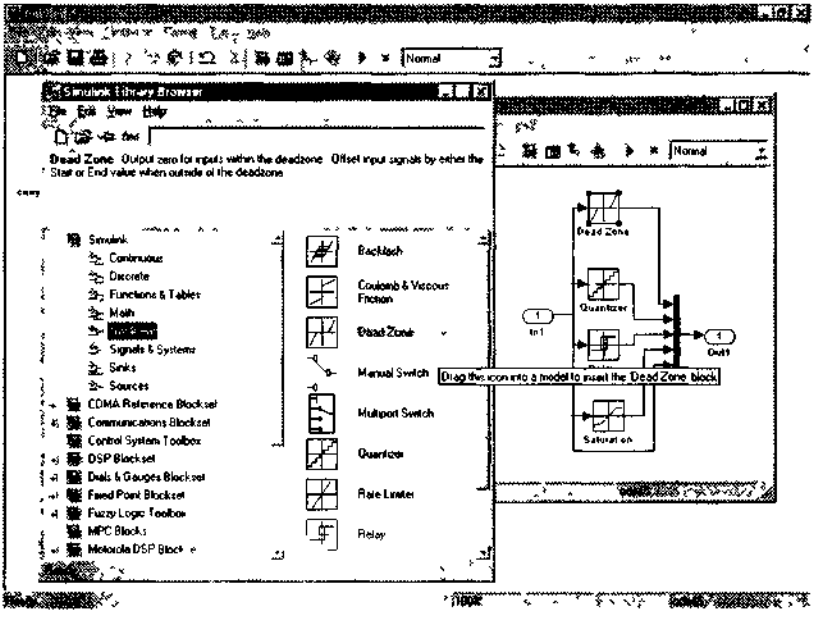


Рис. 9.5. Пример замены одного блока на другой

Браузер удобно использовать для сложных моделей, когда подсистем много и они образуют древовидную структуру. Эту структуру можно наблюдать в левом окне браузера.

Модификация и редактирование подсистемы

Модификация и редактирование подсистемы ничем не отличаются от этих операций для моделей, описанных в главе 4. Поэтому ограничимся простым примером — замены в уже созданной подсистеме одного блока на другой.

Подготовка к удалению блока Backlash показана на рис. 9.4. Намеченный к удалению блок выделяется, и далее используется команда удаления Clear в контекстном меню. Это меню в открытом состоянии показано у блока Backlash. Заметим, что команду Clear можно выполнить и из подменю Edit меню окна подсистемы. После выполнения команды Clear блок Backlash исчезнет и на его место можно ввести новый, например Dead Zone. Это и показано на рис. 9.5.

Нажатием кнопки вызова библиотеки в панели инструментов окна подсистемы вызывается окно библиотеки (рис. 9.5). В нем следует выбрать нужный раздел библиотеки и нужный блок, после чего мышью перенести новый блок на место ранее удаленного блока. Соединения при этом восстанавливаются автоматически, если новый блок позиционирован достаточно точно.

Рисунок 9.6 показывает повторный запуск модели с модифицированной подсистемой.

Таким образом, благодаря применению подсистем можно корректировать последние, не меняя основную модель.

Задание свойств подсистемы

Подсистема имеет окно свойств. Его можно вызвать для просмотра и редактирования с помощью команды меню File ▸ Model Properties окна модели (рис. 9.7).

Окно свойств подсистемы имеет три вкладки:

- Model Properties — свойства подсистемы;
- Options — опции, задающие формат свойств;
- History — данные об истории подсистемы.

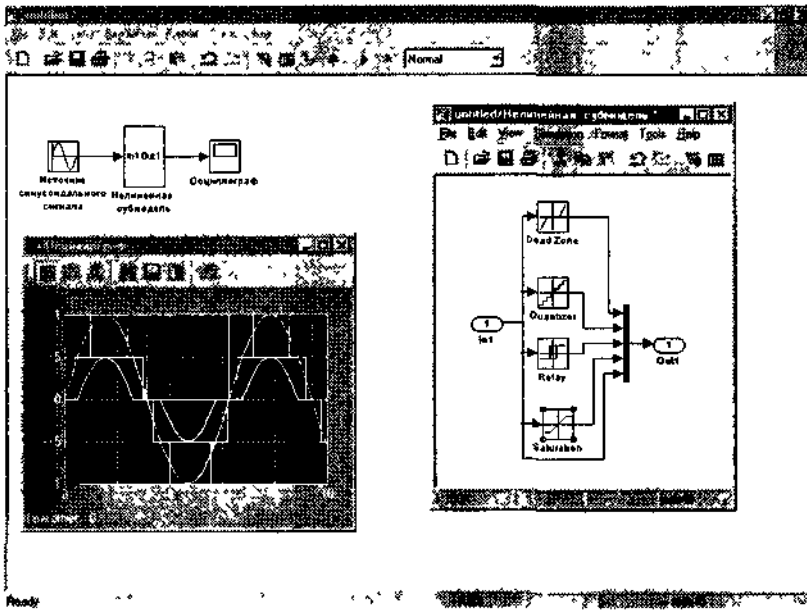


Рис. 9.6. Работа с моделью после коррекции ее подсистемы

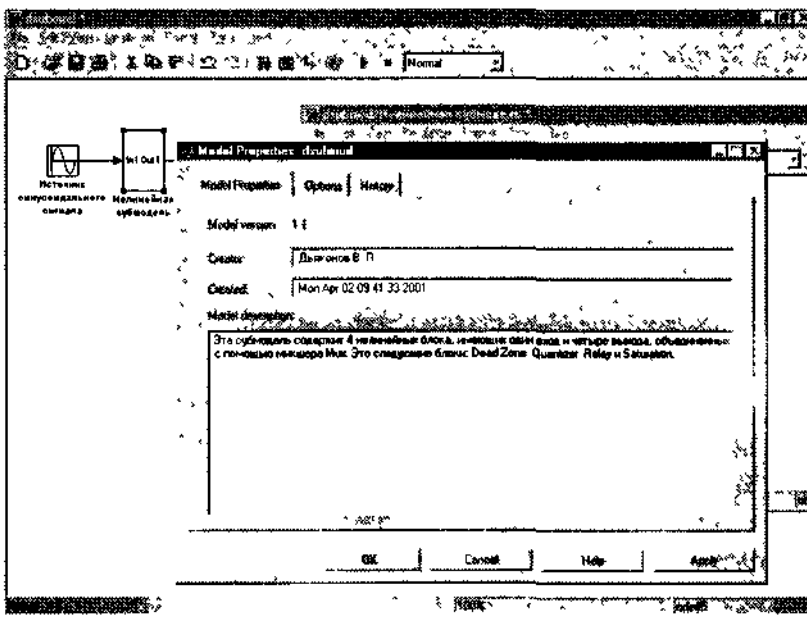


Рис. 9.7. Вызов окна свойств подсистемы

На рис. 9.7 окно свойств подсистемы показано с открытой основной вкладкой. На ней отражается версия подсистемы и дата ее создания. Другие поля вначале пусты. В поле Creator обычно вводится фамилия и инициалы создателя подсистемы, а в поле Model description задается ее описание. После сохранения подсистемы (командой меню File ▸ Save в окне подсистемы) эта информация будет выводиться при вызове окна свойств подсистемы. С содержанием других вкладок пользователь может ознакомится самостоятельно.

К сожалению, окно свойств, оказывается, имеет один и тот же вид для всей системы и для подсистемы. Поэтому следует осторожно относиться к заданию параметров в этом окне — они могут относиться к системе в целом.

Параметры портов ввода и вывода

Порты ввода и вывода являются вполне полноценными блоками и имеют свои окна параметров (рис. 9 8).

Окно порта ввода Inport позволяет задавать следующие параметры:

- Port number — номер порта;
- Port dimension — размерность порта (–1 при динамической установке размерности);
- Sample time — эталонное время;
- Data type — тип данных (выбор из раскрывающегося списка);
- Signal type — тип сигнала (выбор из раскрывающегося списка).

Окно порта вывода Outport задает параметры:

- Port number — номер выходного порта;
- Output when disable — выход при пассивности системы;
- Initial output — инициализация выхода.

Построение подсистем на основе блока SubSystem

Постановка задачи

Предложенный путь создания подсистем путем выделения в их качестве части имеющейся модели не всегда приемлем и не всегда удачен.

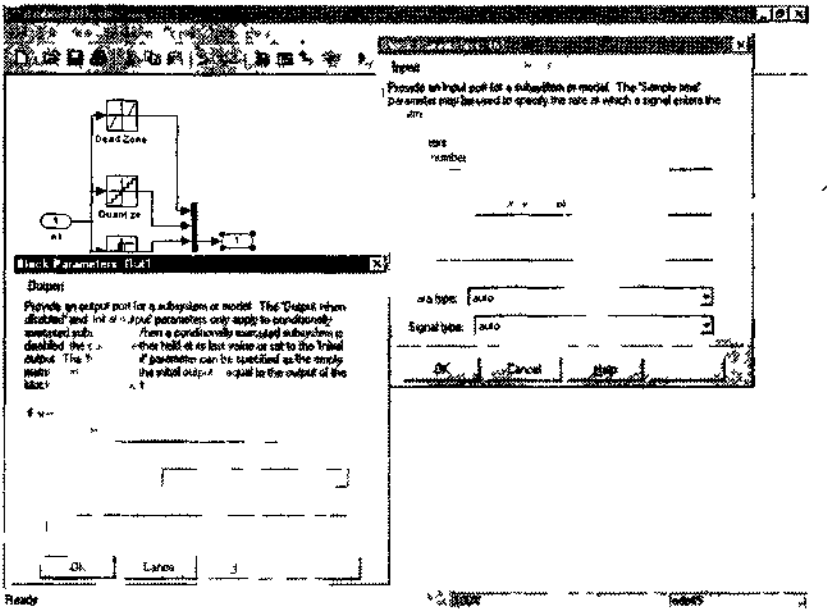


Рис. 9.8. Параметры портов ввода/вывода

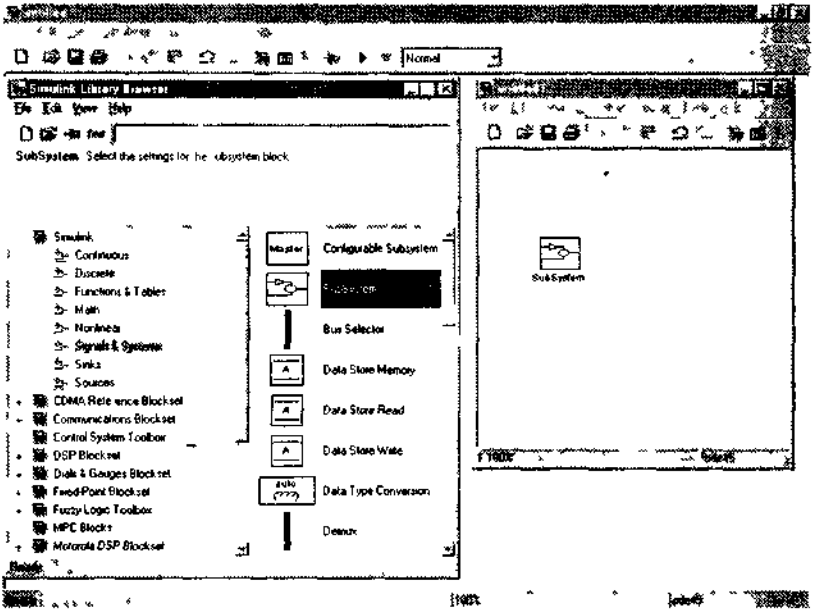


Рис. 9.9. Создание окна будущей модели с блоком подсистемы

Он соответствует подходу «от общего к частному» Но нередко бывает предпочтителен совсем другой подход — «от частного к общему» Применительно к технике моделирования это означает, что сначала создаются подсистемы, а затем уже общая модель системы

Такой подход получил самое широкое признание при создании сложных систем, поскольку он позволяет работать (нередко одновременно) над рядом гораздо более простых систем (подсистем в нашей терминологии) Ниже мы покажем, как этот подход реализуется в пакете Simulink с помощью специального блока SubSystem

Модель функционального генератора

Любопытно, но среди источников сигналов в библиотеке Simulink нет целого ряда широко используемых сигналов, например прямоугольных симметричных разнополярных импульсов типа меандра, треугольных разнополярных импульсов и трапецеидальных импульсов Это сделано, видимо, неслучайно, поскольку средства Simulink позволяют при необходимости создавать источники таких сигналов в виде подсистем

Построим функциональный генератор импульсов, который генерирует три указанные формы импульсов со стандартной единичной амплитудой Трапецеидальные импульсы легко получить, просто ограничив треугольный сигнал Поэтому зададимся целью создать подсистему для генерации меандра и треугольных колебаний

Для генерации меандра достаточно вычесть из стандартного однополярного прямоугольного сигнала (длительность импульса и паузы равны) с амплитудой 2 константу, равную 1

Для получения треугольного сигнала можно использовать интегратор Однако мы получим однополярный треугольный сигнал с амплитудой, вдвое меньшей, чем требуется Чтобы сделать его разнополярным с единичной амплитудой, надо вычесть из него константу 0 5 и «усилить» вдвое масштабирующим блоком Gain

Задание подсистемы с помощью блока SubSystem

Прежде всего откроем пока чистое окно для нашей будущей модели функционального генератора Далее, открыв окно разделов библиотеки Simulink, перетащим мышью в это окно блок SubSystem из раздела Signals&Systems Этот процесс иллюстрирует рис 9 9

Теперь наведем курсор мыши на этот блок и дважды щелкнем мышью. Откроется пустое окно подсистемы (рис. 9.10).

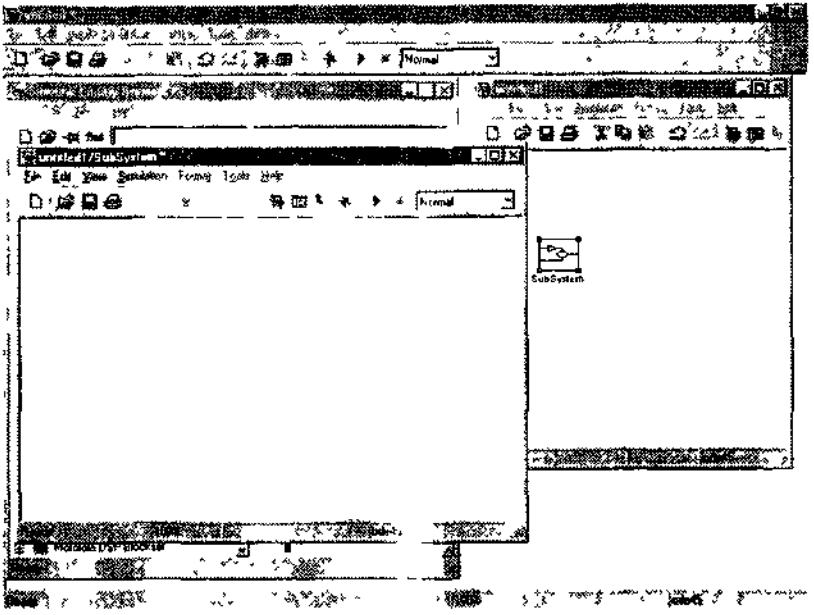


Рис. 9.10. Создание пустого окна подсистемы

Переместив это окно в удобное положение и открыв рядом окно разделов библиотеки Simulink, можно начать строить подсистему в соответствии с описанной выше идеей ее реализации. Рисунок 9.11 показывает почти созданную подсистему генератора меандра и треугольных импульсов. Осталось перенести в подсистему еще один блок выхода. После этого можно будет считать процесс создания подсистемы законченным.

Создание основной модели и ее испытание

Теперь перейдем в окно основной модели. В нем пока представлен только блок SubSystem, созданный выше. Теперь мы можем дополнить общую модель трехвходовым осциллографом и каналом для трапецеидальных импульсов. Этот канал должен содержать ограничитель амплитуды пилообразных импульсов и «усилитель» (блок Gain), доводящий амплитуду трапецеидальных импульсов до заданного уровня 1.0. Рисунок 9.12 показывает созданную таким образом общую модель.

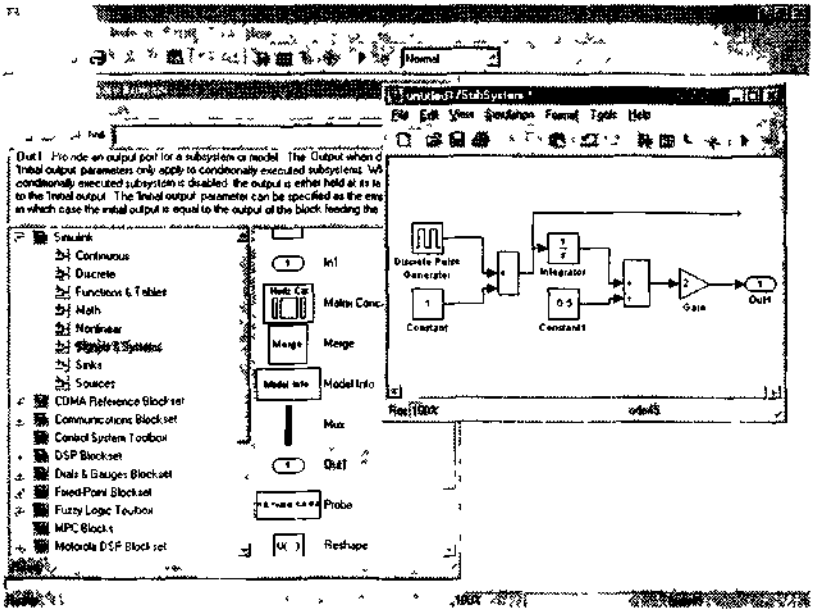


Рис. 9.11. Построение подсистемы

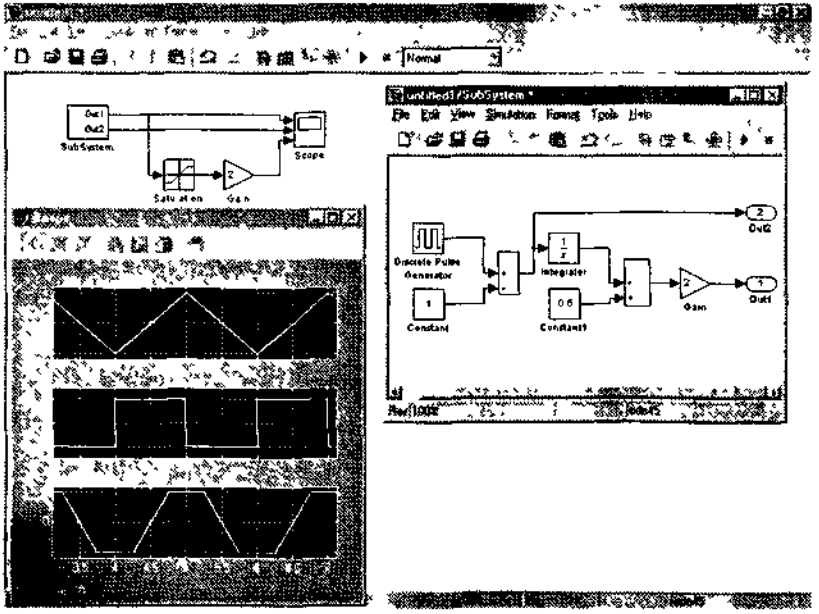


Рис. 9.12. Общая модель функционального генератора

Остается запустить созданную модель. Осциллограммы показывают, что мы полностью достигли цели разработки — наша модель формирует все три типа сигнала.

Управляемые подсистемы

Типы управляемых подсистем

В ряде случаев подсистемы должны быть управляемыми, то есть проявлять активность только при наличии какого-то управляющего события или сигнала. Такие системы называют Conditionally Executed Subsystem (CES). Они обеспечивают упрощение создания сложных систем и дают простое решение задачи синхронизации параллельных процессов.

В Simulink для создания управляемых подсистем служат два блока, расположенных в разделе Signals&Systems. Это блоки включения Enable и триггера Trigger (пускового устройства). Эти блоки можно размещать только в подсистемах — попытка их переноса в окно основной системы приводит к появлению сообщения об ошибке (рис. 9.13).

Рис. 9.13. Окно с сообщением о недопустимости переноса блоков Enable и Trigger в окно основной системы

Наличие в подсистеме указанных блоков является формальным признаком отнесения ее к типу управляемых подсистем.

В зависимости от логики работы управляемых подсистем они делятся на три основных типа:

- E-подсистемы — подсистемы, управляемые блоками Enable;
- T-подсистемы — подсистемы, управляемые блоками Trigger;
- ET-подсистемы — подсистемы, управляемые как блоками Enable, так и блоками Trigger.

Рассмотрим свойства и особенности этих подсистем более подробно.

E-подсистемы

Чтобы сделать подсистему E-подсистемой, достаточно перенести в нее блок Enable и задать (если нужно) его параметры. В качестве

примера превратим нашу подсистему генератора меандра и треугольных колебаний в управляемую E-систему. Для этого перенесем в эту подсистему блок Enable и модернизируем основную модель: уберем канал трапецеидальных импульсов и добавим источник управляющего перепада с задержкой в 4 такта эталонного времени. Полученная новая система показана на рис. 9 14.

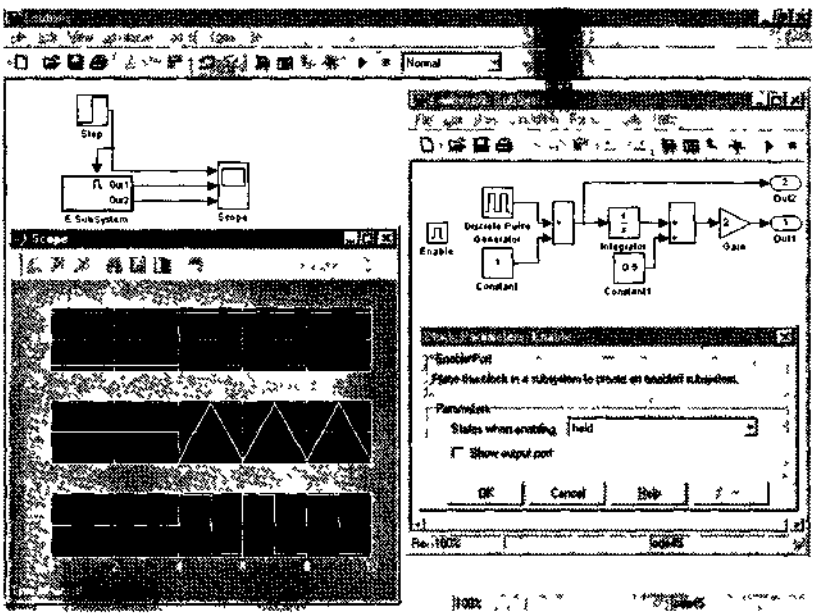


Рис. 9.14. Пример управляемой E-подсистемы

Как видно из осциллограмм рис. 9.14, наша подсистема стала управляемой — теперь импульсы на ее выходе формируются только при подаче на управляющий вход перепада сигнала с выхода источника перепада. Обратите внимание на то, что включение в подсистему блока Enable привело к появлению нового входа на пиктограмме подсистемы, расположенного сверху. Это и есть управляющий вход, и он является внешним признаком того, что подсистема стала управляемой.

Поскольку блок Enable всего лишь задает признак управляемости подсистемы, этот блок никуда не подключается.

Обратите внимание на окно параметров блока Enable, которое показано в окне подсистемы. Это окно содержит единственный параметр

States when enabled, задающий состояние подсистемы перед ее запуском, значение которого выбирается из раскрывающегося списка:

- held (сохранение) — применение предшествующего состояния,
- reset (сброс) — применение начального состояния (при инициализации).

Кроме того, если установлен флажок Show output port, то блок Enable приобретает выходной порт, который можно использовать для управления другими подсистемами или блоками. В пассивном состоянии подсистемы сигнал на выходе выходного порта равен 0, а при активной равен 1. Рисунок 9.15 иллюстрирует применение дополнительного выходного порта подсистемы.

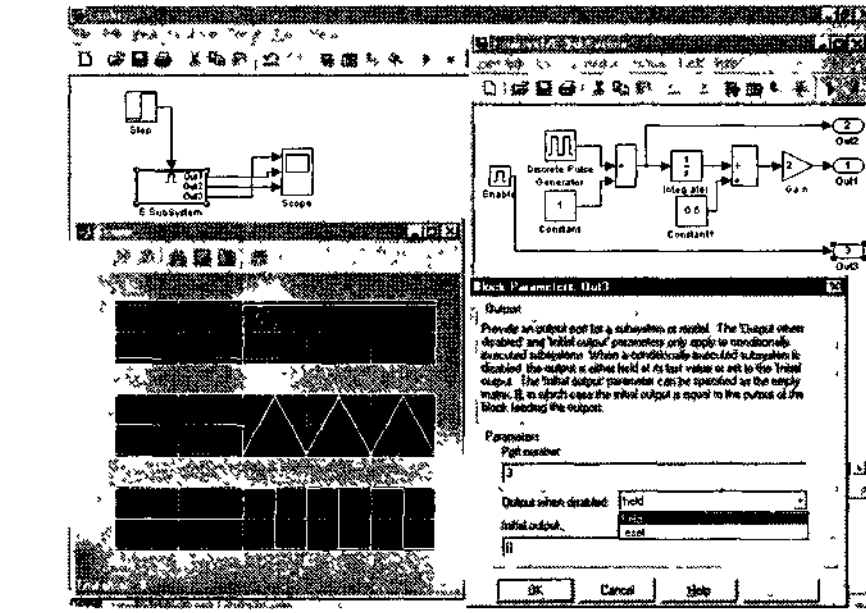


Рис. 9.15. Пример использования выходного порта блока Enable управляемой E-подсистемы

Обратите внимание на то, что в данной системе сигнал от управляющего перепада уже не подается явно на верхний вход осциллографа. Теперь на осциллограф подан сигнал с выходного порта блока Enable, подключенного к выходному порту Out3 управляемой подсистемы. На рис. 9.15 под подсистемой показано окно параметров ее выходного порта Out3.

T-подсистемы

Для придания подсистеме статуса T-системы достаточно ввести в нее блок триггера Trigger. После этого подсистема с определенными ограничениями (они отмечены ниже) становится управляемой по триггерному входу. Управление происходит по перепаду управляющего сигнала.

Работа T-подсистем имеет ряд отличительных признаков:

- подсистема работает только на том шаге, на котором имел место перепад управляющего сигнала;
- подсистема не возвращается в исходное состояние, и ее текущее состояние сохраняется до очередного запуска,
- отсутствует параметр Output when disabled у блока выхода;
- возможны различные виды обозначения входного порта управляющего сигнала;
- в блоках подсистемы, имеющих параметр эталонного времени Sample time, следует задавать этот параметр равным –1.

Большинство этих признаков носит существенный ограничительный характер. К сожалению, последний признак нередко присутствует даже в неявном виде. Например, для аналоговых интеграторов не задается параметр Sample time в явном виде, но фактически этот параметр отличен от –1. Поэтому попытка напрямую использовать интегратор в T-подсистеме обычно обречена на провал.

Пример применения T-подсистемы

Прекрасный пример построения T-подсистем имеется среди новых демонстрационных примеров пакета Simulink 4. Он представлен на рис. 9.16.

В этом примере заданы три подсистемы-«пустышки». Они соответствуют трем типам T-подсистем с разными вариантами управления. Обозначения таких подсистем, их графическое представление и окно параметров блока Trigger одной из подсистем показаны на рис. 9.16.

Казалось бы, какой смысл в подсистеме-«пустышке», в которой нет ничего, кроме входного и выходного портов, соединенных напрямую друг с другом? Однако следует помнить, что такая T-подсистема активна, то есть работает как короткозамкнутая перемычка, толь-

ко тогда, когда это разрешает управляющий порт (вход) Таким образом, она превращается в управляемый ключ, логика которого определяется параметром Trigger type

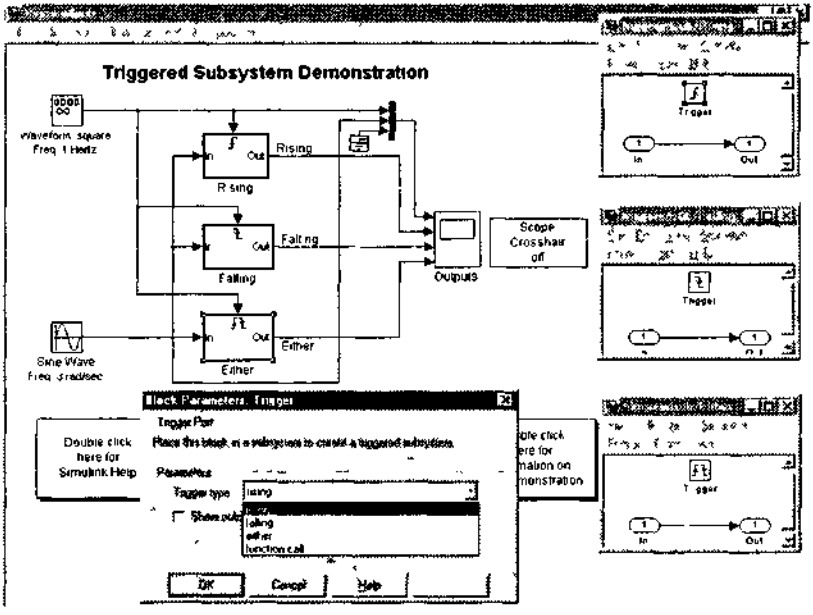


Рис. 9.16. Пример применения простейших Т-подсистем

Этот параметр может иметь следующие значения:

- rising — срабатывание при нарастании сигнала управления;
- falling - срабатывание при спаде сигнала управления,
- either -- срабатывание как при нарастании, так и спаде управляющего сигнала;
- function-call – срабатывание по логике заданной S-функции, в этом случае управляющий вход имеет обозначение f().

Этот пример интересен тем, что в нем используются все три возможных типа сигналов управления с внутренней логикой. Создаваемые этой моделью (с ее подсистемами) сигналы представлены на рис 9.17. Генерация сигналов начинается в момент пуска моделирования Это важно для понимания отличия Т-подсистем от ЕТ-подсистем, описанных далее.

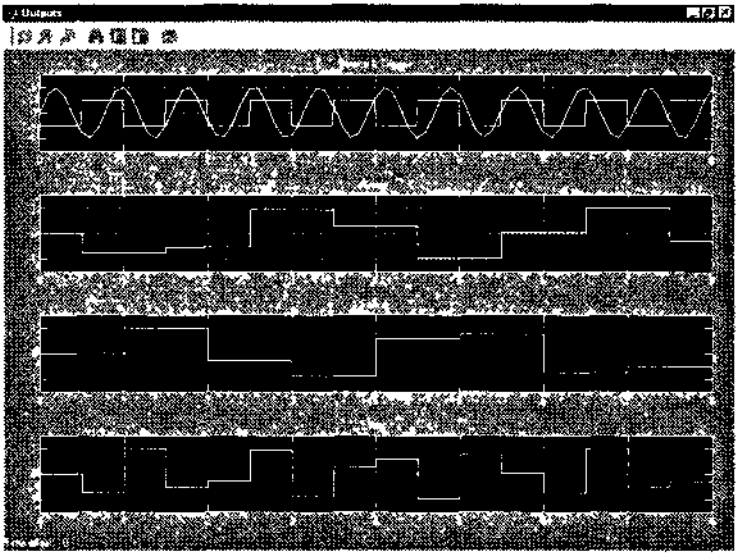


Рис. 9.17. Осциллограммы модели рис 9 16

ET-подсистемы

ET-подсистемы — это подсистемы, использующие одновременно описанные выше механизмы управления, реализованные блоками Enable и Trigger При этом на применение таких подсистем оказывают влияние ограничения, присущие как E-подсистемам, так и T-подсистемам.

Для наглядной иллюстрации создания ET-подсистем потребуем от модели рис. 9 16, чтобы первая и третья подсистемы стали ET-подсистемами Для этого дополним их блоками Enable. У этих подсистем появится второй вход, на который можно подать, например, разрешающий сигнал от генератора перепада (рис 9.18). Таким образом мы обеспечим генерацию сигналов на выходе первой и третьей подсистемы только при наличии сигнала на их дополнительных входах

Рисунок 9.19 показывает осциллограммы выходных импульсов новой модели Как и следовало ожидать, сигналы на выходах первой и третьей подсистем появляются спустя время задержки перепада (4 такта времени Sample time) Сигнал на выходе второй подсистемы (оставленной для сравнения в неизменном виде) появляется сразу после запуска модели

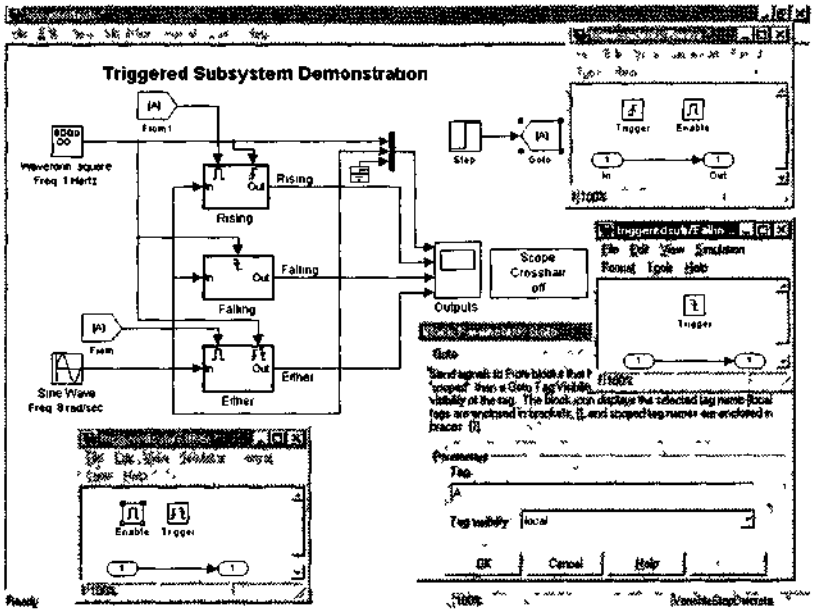


Рис. 9.18. Пример системы с одной Т-подсистемой и двумя ET-подсистемами

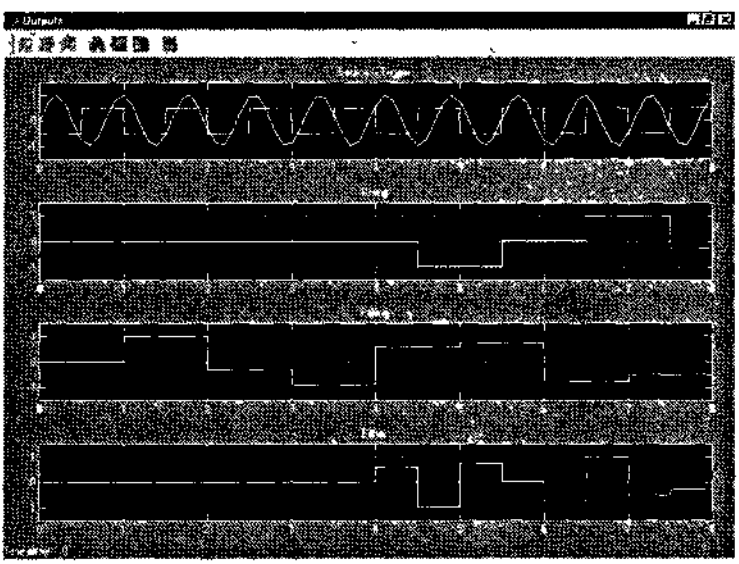


Рис. 9.19. Осциллограммы модели, представленной на рис 9 18

Применение блоков Goto, Goto Tag visibility и From

Чем сложнее моделируемая система, тем труднее представить ее графическое изображение, особенно если желательно, чтобы оно не выходило за пределы окна модели, видимые на экране монитора. В подобном случае весьма полезными являются блоки Goto и From, позволяющие создавать в моделях (и в подсистемах) невидимые связи.

Применение этих блоков показано на рис. 9.18. Здесь блок Goto подключен к выходу генератора управляющего перепада, а блоки From и From1 подключены к входам E-управления первой и третьей подсистем. Таким образом, управляющий перепад подается на эти входы без использования дополнительных соединений.

На рис. 9.18 показано также окно параметров блока Goto — передатчика сигнала на блоки From. В этом окне имеются два параметра:

- Tag — имя (метка) передаваемых данных;
- Tag visibility — область видимости передаваемых данных.

Блок From имеет единственный параметр — Tag.

С позиции применения языков программирования блоки From и Goto подобны процедурам и обращениям к ним. Такое представление достаточно естественно, если учесть, что Simulink является средством визуального программирования. Кроме того, такое представление делает достаточно очевидным смысл параметра Tag visibility блока Goto, который используется для передачи данных с установленной областью видимости и может принимать следующие значения:

- local — локальные данные, доступные только той подсистеме, в которой имеется блок Goto;
- scoped — данные, область действия которых распространяется на все подсистемы более низкого уровня;
- global — глобальные данные, доступные всем подсистемам и основной системе.

Нетрудно заметить, что смысл этого параметра очень напоминает смысл статуса локальных и глобальных переменных во многих языках программирования.

Глава 10. Создание собственных блоков и библиотек

- Маскированные подсистемы
- Расширенные средства создания пиктограмм
- Создание библиотек пользователя

Маскированные подсистемы

Механизм маскирования

Пользователь, всерьез занявшийся моделированием систем и устройств, рано или поздно сталкивается с необходимостью подготовки собственных блоков, обладающих свойствами стандартных библиотечных блоков пакета Simulink. Здесь надо отметить, что сами по себе подсистемы, описанные в предшествующем уроке, такими качествами не обладают. Главное отличие подсистем от блоков в том, что подсистемы не имеют ни своей уникальной пиктограммы, ни окна параметров и не связаны с разделом библиотеки.

Для построения пользовательских блоков Simulink предлагает специальный механизм маскирования подсистем. Маскированные подсистемы — это такие подсистемы, которые имеют специальный признак (маску), скрывающий их внутреннюю, иногда достаточно сложную, структуру. В результате такая подсистема в деталях не видна и воспринимается как библиотечный модуль Маскированные подсистемы обладают рядом важных достоинств:

- они имеют свои пиктограммы с уникальными изображениями;
- их можно использовать как библиотечные блоки;
- у них есть свое окно установки параметров;
- есть возможность в любой момент сбросить маску и наблюдать структуру блока;

- применение масок расширяет возможности построения сложных моделей,
- имеется возможность легко отредактировать подсистему, превращенную в маскированную;
- повышается наглядность моделей-диаграмм;
- повышается защищенность подсистемы от модификации, в том числе преднамеренной.

Для создания маскированных подсистем надо выполнить следующие операции:

- разработать и протестировать модель с соответствующими блоками;
- выделить часть блоков и оформить их в виде подсистемы (см. главу 9);
- задать подсистеме статус маскированной подсистемы;
- с помощью специального редактора масок создать окно установки параметров, документацию под маскированную подсистему и ее справочную систему;
- выполнить (если необходимо) тестирование основной модели с ее маскированными подсистемами и прочими блоками;
- выполнить (если необходимо) редактирование созданной маскированной подсистемы;
- демаскировать (если необходимо) подсистему.

Большая часть этих операций выполняется с помощью редактора маскированных подсистем, или просто масок.

Создание начальной модели

Рассмотрим следующий пример. Пусть нам надо создать модель, выполняющую функцию вычисления значения $y = ax^2 + b$, где a и b — константы. Для возведения x в квадрат достаточно использовать умножитель, на оба входа которого подан сигнал x. Подключив к выходу умножителя масштабирующий блок Gain с коэффициентом передачи a, мы получим сигнал ax^2. К этим блокам нам необходимо добавить сумматор, на один вход которого надо подать сигнал с умножителя, а на другой — выход константы b. В итоге на его выходе будем иметь сигнал $ax^2 + b$, что и требовалось.

Такую функцию реализует модель, представленная на рис. 10.1. На этом рисунке показаны осциллограммы входного синусоидального

сигнала и выходного сигнала при следующих параметрах: коэффициент передачи блока Gain равен 1 и константа b равна 0.

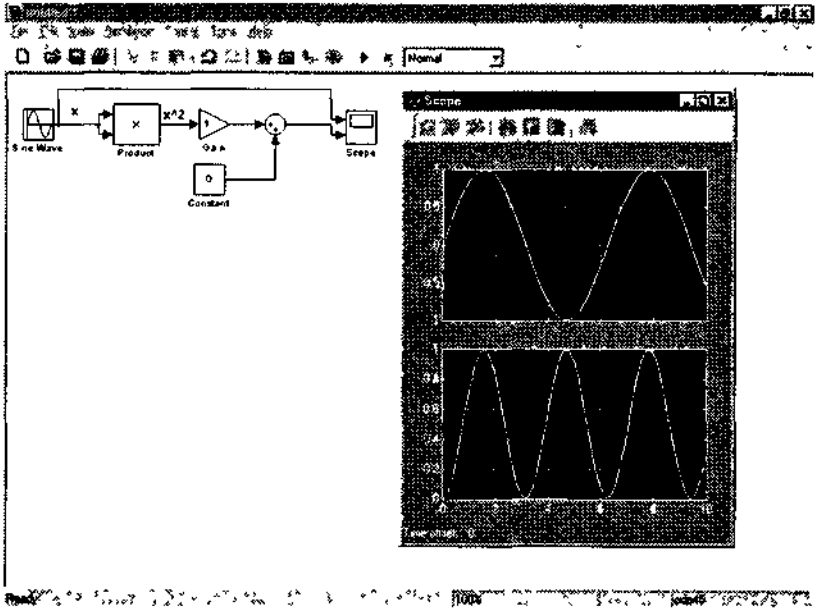


Рис. 10.1. Исходная модель

Любопытно, что выходной сигнал очень напоминает синусоиду, но удвоенной частоты и с добавлением постоянной составляющей. Это говорит о том, что с помощью подобного устройства в принципе легко создать широкодиапазонный удвоитель частоты, что представляет определенный интерес для радиотехники, где умножители частоты применяются довольно широко.

Подготовка к маскированию подсистемы

Из исходной модели хорошо видно, что, исключив из нее источник синусоидального сигнала и осциллограф, мы получим функционально законченный блок. Назовем его квадратором (Quadrator) и приступим к его маскированию.

Первое, что необходимо сделать, — это перейти от конкретных параметров блоков к обобщенным. Так, мы должны задать коэффициент передачи масштабирующего блока Gain равным не 1, а a (a —

переменная, которая впоследствии будет принимать любые значения). Далее вместо константы 0 у блока Constant надо задать значение *b*. Кроме того, надо выделить отведенные под подсистему блоки. Все это показано на рис. 10.2

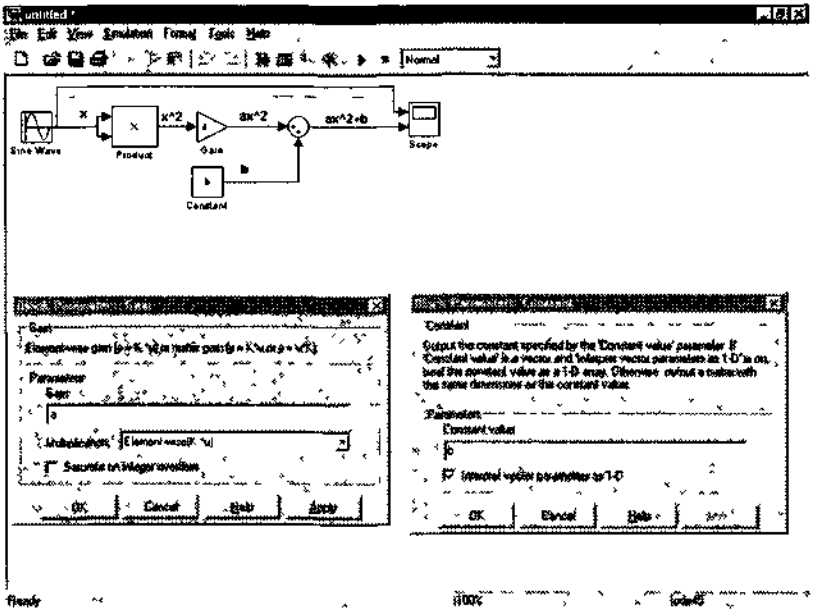


Рис. 10.2. Подготовка к созданию подсистемы

Выполнив команду меню Edit ▸ Create Subsystem, создадим на базе выделенных блоков подсистему. Особенности этого процесса детально описывались в главе 9, и мы не будем их повторять. Отметим лишь, что если выполнить двойной щелчок мышью на блоке подсистемы, то появится окно с графической моделью подсистемы (рис. 10.3).

Запуск редактора маски

Для создания маски достаточно выделить нужную подсистему, установив на ней курсор мыши и щелкнув ее левой кнопкой, после чего выбрать команду Mask Sybsystem... меню Edit.

Данная команда запускает редактор маски — появляется его окно с открытой незаполненной вкладкой Icon (рис. 10.4).

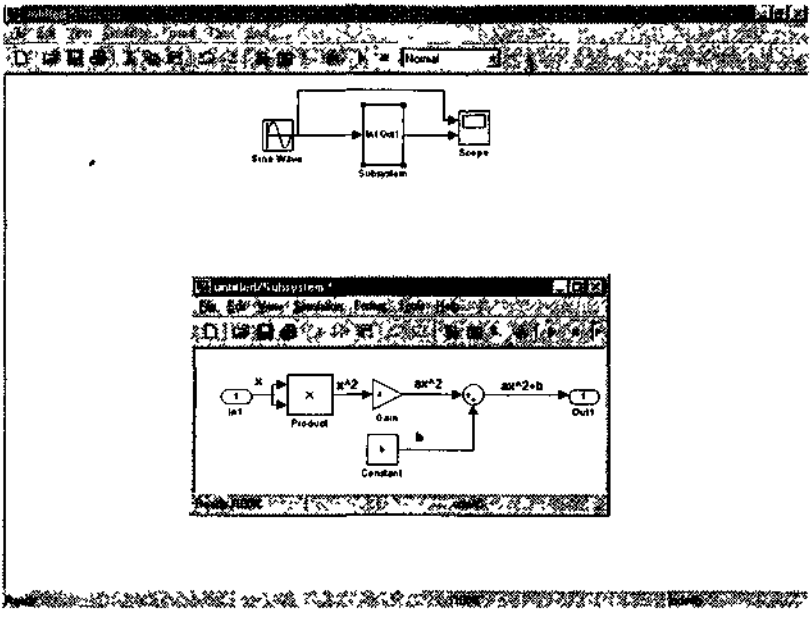


Рис. 10.3. Преобразованная модель с подсистемой

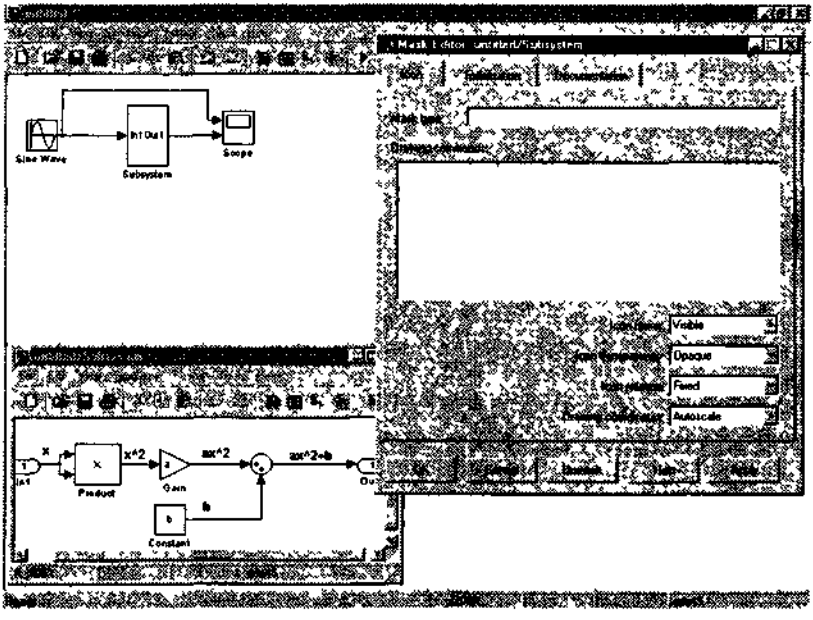


Рис. 10.4. Окно редактора маски после его запуска

Описание редактора маски

Как видно из рис. 10 4, редактор маски имеет три вкладки. Отметим их назначение:

- Icon — подготовка пиктограммы (значка) блока;
- Initialization — подготовка окна задания параметров блока;
- Documentation — подготовка документации по блоку.

Сверху окна имеется общее для всех вкладок поле Mask type, в которое заносится имя блока. Внизу имеются также общие для всех вкладок пять кнопок:

- OK — завершить установки и создать маску;
- Cancel — отказаться от создания маски и закрыть окно редактора;
- Unmask — снять маску с выделенного блока;
- Help — открыть окно справки по редактору маски;
- Apply — применить заданные установки.

Следует отметить, что после создания маски команда Undo не работает, так что в процессе задания параметров маски не стоит нажимать клавишу OK. Делайте это, когда вы уверены, что все данные для маскируемой системы введены. В ходе ввода данных целесообразно пользоваться клавишей Apply. Она вводит заданные данные, и некоторые из них тут же отражаются на виде пиктограммы маски.

Создание окна параметров блока

Как отмечалось, окно редактора маски появляется с открытой вкладкой Icon. Однако большинство пользователей считают создание пиктограммы блока (или его значка) вовсе не первоочередным делом. Важнее создать окно параметров блока. Оно может содержать параметры блока, а также флажки. Для нашего блока Quadrator надо создать окно с двумя параметрами — *a* и *b*. Редактор маски позволяет задать до 14 параметров и элементов управления, причем разного вида (см. ниже).

Для создания данных окна параметров служит вкладка Initialization (рис. 10.5).

При первом открытии вкладка Initialization содержит пустой список блоков Prompt. В нем присутствует только строка <<end parameter list>> (конец списка параметров). Этот список состоит из трех столбцов:

- Prompt — имена используемых в маске блоков;
- Type — типы параметров (выбираются из раскрывающегося списка Control type);
- Variable — имена переменных, несущих значение соответствующих параметров.

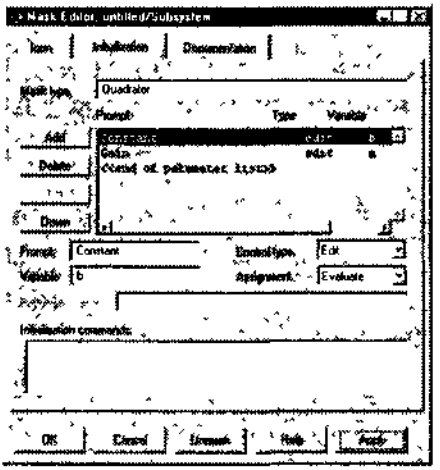


Рис. 10.5. Подготовка данных для окна задания параметров блока

Для создания этого списка служат четыре кнопки вкладки:

- Add — добавить в список новую позицию;
- Delete — стереть выделенную позицию списка;
- Up — поднять выделенную позицию списка вверх;
- Down — опустить выделенную позицию списка вниз.

Ниже находятся поля Prompt для задания имени блока, к которому относится задаваемый параметр, и Variable для задания имени переменной, которая будет нести значение параметра. В нашем случае, нажав кнопку Add, следует задать сначала имя блока Gain и переменную *a*, а затем имя блока Constant и переменную *b*. Для завершения этого процесса достаточно нажать кнопку Apply.

Дополнительные возможности задания параметров

Вкладка Initialization имеет еще два раскрывающихся списка:

- Control type — определяет тип элемента интерфейса окна параметров;

- Assignment — выбор типа параметра (Evaluate — если параметр является вычисляемым и имеет числовое значение, Literal — если параметр имеет строковый формат).

Список Control type позволяет задать следующие типы элементов интерфейса в будущем окне задания параметров блока:

- Edit — обычное поле для ввода параметра;
- Checkout — флажок;
- Popup — раскрывающийся список.

Со всеми этими элементами интерфейса мы уже многократно сталкивались — см., например, окна параметров, показанные на рис. 2.3. Поэтому более детальное описание их не требуется.

На вкладке Initialization есть еще два поля:

- Popup strings — ввод элементов раскрывающегося списка;
- Initialization command — ввод списка команд инициализации маски.

Поле Popup strings доступно для элементов типа Popup. Вводимые элементы списка должны разделяться знаком | (вертикальная черта).

Поле Initialization command содержит заданные по правилам языка программирования MATLAB команды инициализации параметров. Например, если мы хотим задать начальные значения параметров $a = 1$ и $b = 0$, то в это поле мы должны записать строки `a = 1` и `b = 0`.

Итак, мы рассмотрели все возможности задания элементов будущего окна параметров маскируемой подсистемы. В дальнейшем их можно отредактировать, если возникнет необходимость изменить интерфейс окна параметров блока.

Подготовка описания и документации блока

Библиотечные блоки Simulink имеют два основных типа описания:

- описание блока, размещенное вверху окна установки параметров;
- справочное описание блока, размещаемое в справочной системе.

Вкладка Documentation позволяет создавать описания обоих типов. Пример описания для создаваемого нами блока дан на рис. 10.6.

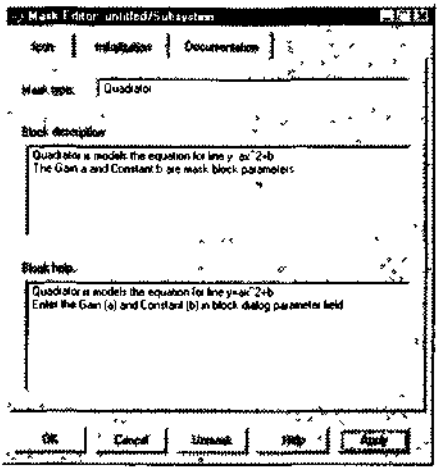


Рис. 10.6. Подготовка описания и справки по блоку

Поле Block Description (Описание блока) служит для ввода текстового описания блока, которое будет размещено вверху окна параметров блока. Его размером не стоит злоупотреблять.

Поле Block Help (Справка по блоку) служит для ввода текстовой части справки, которая будет размещена в справочной системе Simulink. Справка должна быть достаточно подробной.

Таким образом реализуются единообразные правила описания блоков, созданных пользователем, и стандартных библиотечных блоков. Завершив ввод описаний, следует нажать кнопку Apply в окне редактора масок.

Создание простой пиктограммы блока

Теперь настала пора подумать о том, какой конкретно будет пиктограмма (значок) нашего блока. Для ее подготовки надо открыть вкладку Icon окна редактора маски. Рисунок 10.7 поясняет создание простейшей пиктограммы, которая представляет собой прямоугольник с надписью внутри, состоящей из двух строк: Quadrator и y=ax^2+b.

Как видно из рис. 10.7, вкладка Icon содержит поле Drawing command, в котором задаются текстовые и графические команды создания пиктограммы. Они задаются по правилам языка программирования MATLAB. Так, для создания указанной пиктограммы использована команда вывода текста посередине графического окна disp('Текст').

Чтобы разбить строку текста (в нашем случае Quadrator и y=ax^2+b) на две строки, использован специальный символ перевода строки \n.

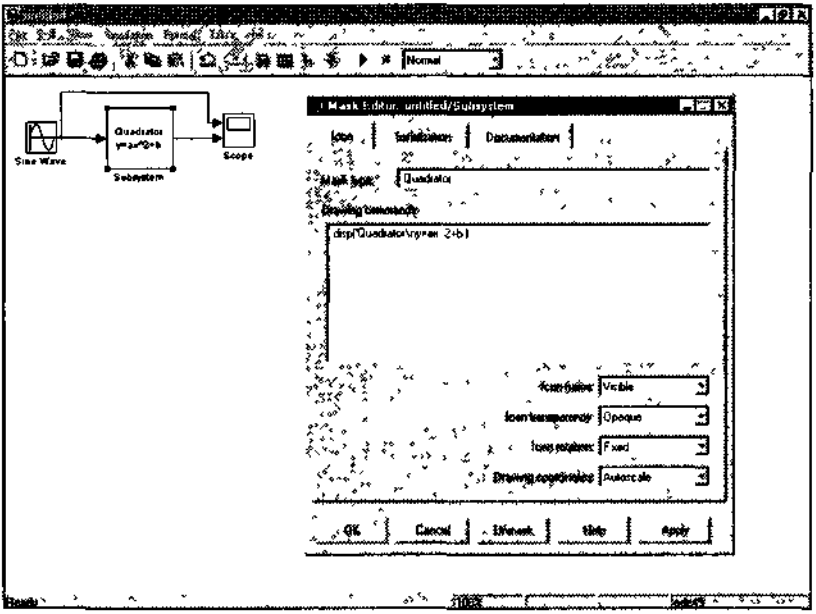


Рис. 10.7. Создание простой пиктограммы

Пока для задания пиктограммы нашего блока этого достаточно. Нажав кнопку Apply, можно наблюдать превращение нашего блока в блок с новой пиктограммой (см. рис. 10.7 сверху). Пока наш блок имеет снизу название Subsystem.

Нам осталось завершить процесс создания маски нажатием клавиши OK. На практике перед этим рекомендуется еще раз просмотреть все установки на всех вкладках и скорректировать замеченные неточности, если таковые есть. Нажав клавишу OK, мы фиксируем создание маскированной подсистемы.

Проверка модели с созданной маской

Если выполнить двойной щелчок мыши на созданной маскированной подсистеме (маске), то появится окно параметров маски (рис. 10.8).

Запустив модель кнопкой пуска моделирования, можно наблюдать работу модели и, в частности, появление осциллограмм. Сравнение их с осциллограммами исходной модели (рис. 10.1) показывает их

полную идентичность, что свидетельствует о том, что созданная нами маскированная подсистема работает так, как это было задумано.

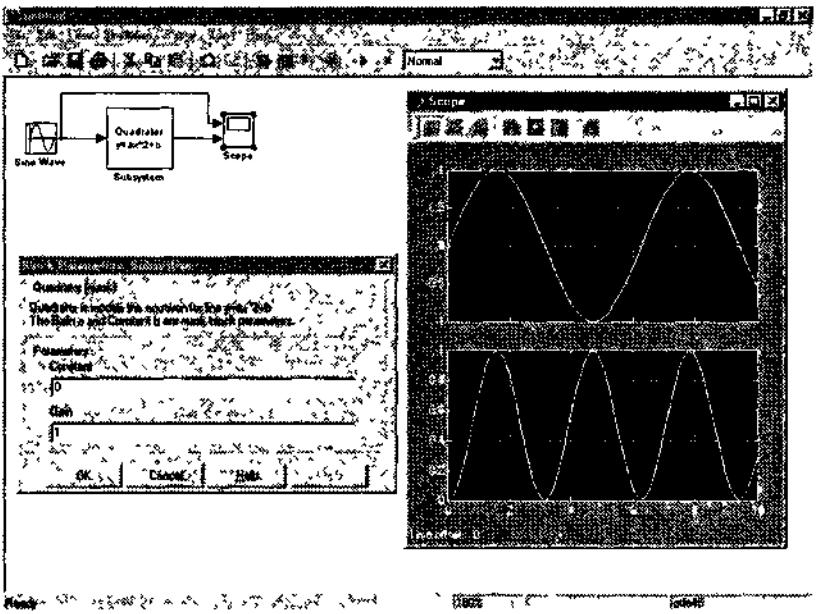


Рис. 10.8. Работа с моделью, имеющей маскированную подсистему (маску)

Вывод описания и справки маски

В верхней части окна параметров созданной маскированной подсистемы можно увидеть ранее созданное описание блока. А если нажать мышью кнопку Help в этом окне, то появится раздел стандартной справочной системы Simulink с текстом также ранее введенной справки (рис. 10.9)

Итак, созданная маскированная подсистема приобрела все атрибуты библиотечного блока. Впрочем, одно важное отличие есть: в то время как библиотечные модули защищены от модернизации, маскированные модули остаются доступными для редактирования. Более того, их легко можно демаскировать.

Маски-справки

На диаграммах моделей часто можно увидеть блоки со справочными данными (см , например, два таких блока под диаграммой на

рис. 2.7). Их активизация открывает окно с текстом справки или описанием диаграммы. Вы также можете создать такие блоки в виде масок. Они могут иметь пиктограмму в виде вопросительного знака или текста с пояснением правил работы с такой маской.

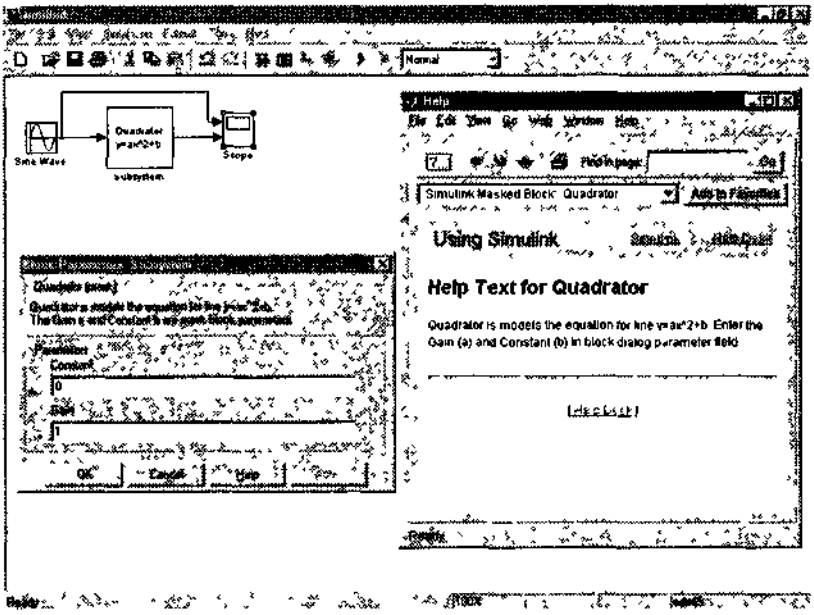


Рис. 10.9. Пример вывода описания и справки по маске

Обычно двойной щелчок мышью на пиктограмме такого блока открывает его окно и выводит текст справки. Можно также задать специальную функцию открытия, описав ее в поле Open function окна свойств блока Block Properties. Однако эта возможность используется довольно редко.

Расширенные средства создания пиктограмм

Задание текстовых надписей

Simulink вместе с базовой системой MATLAB открывает огромные возможности в создании любых пиктограмм. Но для этого необхо-

димо использовать язык программирования системы MATLAB для вывода текстов и построения графиков. Учитывая ограниченный объем книги и ее направленность, мы воздержимся от подробного описания этих средств. Дадим только те их них, которые образуют «золотую середину» в части возможностей оформления пиктограмм

Прежде всего отметим наиболее важные команды задания текстовых надписей в пиктограммах:

- `disp('Текст')` — задание надписи Текст в середине значка;
- `text(x,y,'Текст')` — задание надписи Текст с началом в точке с координатами (x, y);
- `fprint('Текст')` — вывод форматированного текста по центру пиктограммы.

В этих командах текстовая строка задается явно или в виде значения строковой переменной, например `disp(var)`, где `var='Текст'`. Для перевода части текста на новую строку используется комбинированный символ перевода строки `\n`.

Здесь также уместно отметить, что можно легко сменить текстовую подпись маскированной подсистемы. Вместо стандартной надписи Subsystem можно ввести задуманное имя блока — Quadrator.

Применение команд графики MATLAB

На пиктограммах часто встречаются графические изображения. Например, для пиктограммы блока какой-либо функции нет ничего лучшего, чем изобразить хотя бы упрощенный график этой функции. Для этого можно использовать графическую команду `plot(X,Y)`, которая строит линию из отрезков прямых, соединяющих узловые точки с координатами, заданными в векторах X и Y.

Зададимся целью создать пиктограмму блока Quadrator в виде прямоугольника с графиком параболы и надписью в ней `y = ax^2 + b` Для этого, выделив блок, выполним команду меню File ▸ Edit ▸ Mask (Редактирование маски) в окне моделей Simulink. В появившемся окне редактора маски на вкладке Icon в поле Drawing commands зададим нужные команды (рис. 10.10).

Подбирая координаты точек в команде plot опытным путем, можно после нескольких попыток получить требуемое изображение параболы. Фиксируйте изменения нажатием клавиши Apply и наблюдайте за построениями.

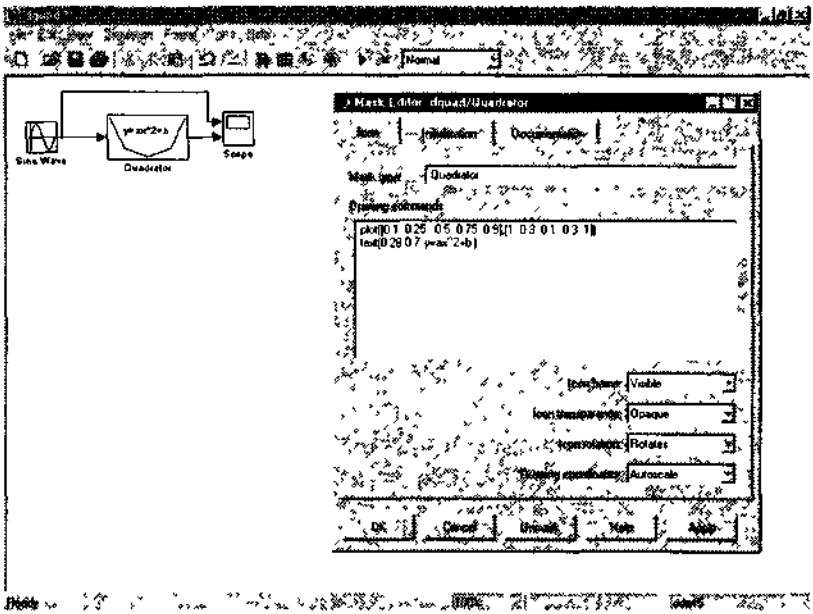


Рис. 10.10. Пример редактирования пиктограммы блока

Средства специального оформления пиктограмм

На вкладке Icon мы пока не использовали четыре важных элемента — раскрывающиеся списки:

- Icon frame — тип отображения рамки пиктограммы:
  - Visible — рамка видна;
  - Invisible — рамка не видна.
- Icon transparency — задание прозрачности пиктограммы:
  - Opaque — пиктограмма непрозрачна;
  - Transparent — пиктограмма прозрачна.

Если пиктограмма прозрачна, то через новое изображение будет просматриваться старое. Иногда это бывает полезно: например, если старое изображение содержит входные и выходные порты и их подписи, то они будут видны на новой пиктограмме.

- Icon rotation — задание возможности вращения пиктограммы:
  - Rotate — пиктограмма может вращаться.

- Fixed — ее положение фиксировано.

- Drawing coordinates — задание условий масштабирования и типа графики:
  - Autoscale — автоматическое масштабирование;
  - Normalized — нормализованное масштабирование;
  - Pixel — представление графики в пикселах.

Пользователь может легко опробовать действие параметров, вводимых этими раскрывающимися списками. Например, на рис. 10.11 показан пример поворота на 90 градусов пиктограммы, созданной ранее. При таком повороте потребуется скорректировать связи между блоками.

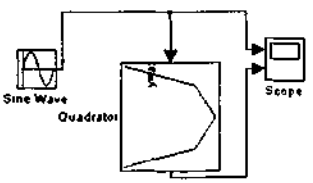


Рис. 10.11. *Пример поворота пиктограммы*

Успех нашего опыта на рис. 10.11 лишь частичный. Нетрудно заметить, что поворот рисунка прошел гладко, но вот надпись явно «заехала» куда-то не туда (как решить эту проблему, см. ниже).

Рассмотрим последний раскрывающийся список — Drawing coordinates. Каждый из элементов этого списка имеет свои особенности. Параметр Autoscale позволяет автоматически менять масштаб при растяжении пиктограмм в разных направлениях. Однако это относится только к графике и не касается текстовых надписей. Именно поэтому надпись на рис. 10.11 «уплыла».

Параметр Normalized задает нормализованное изображение. При этом левый нижний угол окна пиктограммы имеет координаты (0, 0), а правый верхний — (1, 1). Это облегчает задание параметров графических команд, значения которых не должны выходить за пределы [0,1].

Параметр Pixel используется при задании параметров графических команд в пикселах. В этом случае решение задач (довольно непростых) по изменению масштаба и поворота рисунков пиктограмм возлагается целиком на пользователя.

Применение графического редактора пиктограмм

MATLAB имеет специальный графический редактор для построения простых рисунков из отдельных линий. Этот редактор запускается прямо из командной строки MATLAB командой `iconedit` (рис. 10.12).

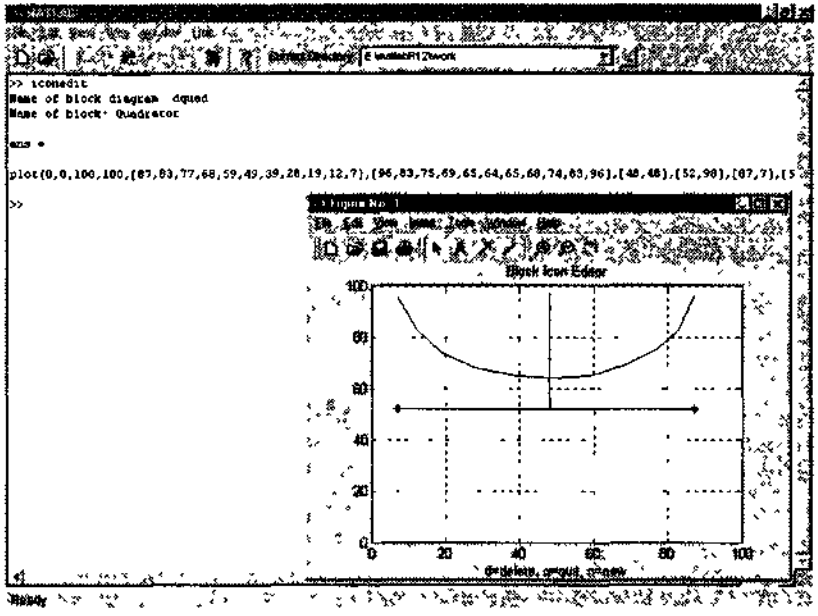


Рис. 10.12. Пример работы с графическим редактором пиктограмм

При этом запрашивается имя диаграммы модели и блока, для которого создается пиктограмма. Этот диалог виден на рис. 10.12. Можно также выполнить команду `iconedit('Maska', 'Subsystem')`. В этом случае диалог при запуске отсутствует. В том и в другом случае появляется окно редактора пиктограмм (рис. 10.12).

Работа с редактором элементарна. Сначала имеется пустое окно редактирования рисунка. Для создания рисунка по точкам используется мышь и всего три команды, вводимые с клавиатуры:

- `d` — удаление последней точки;
- `n` — создание новой точки для построения новой линии;

- q – выход из редактора с автоматическим обновлением рисунка пиктограммы.

При помощи мыши можно перемещать крестообразный графический курсор. При нажатии левой кнопки мыши строится очередная точка и соединяется с предшествующей точкой. Указанные выше команды позволяют удалять ошибочно введенные точки, создавать новые точки для построения новых линий и завершить работу с редактором. На рис. 10.13 показан пример построения параболы с координатными осями.

По завершении работы с редактором пиктограмм его окно закрывается, и в командной строке MATLAB появляется графическая команда, обеспечивающая построение заданного графика. Ее можно (командой Edit ▸ Copy) поместить в буфер, а затем командой Edit ▸ Paste в окне редактора маски перенести в поле Drawing commands вкладки Icon. Можно также добавить команду вывода текстового комментария. Все это показано на рис. 10.13.

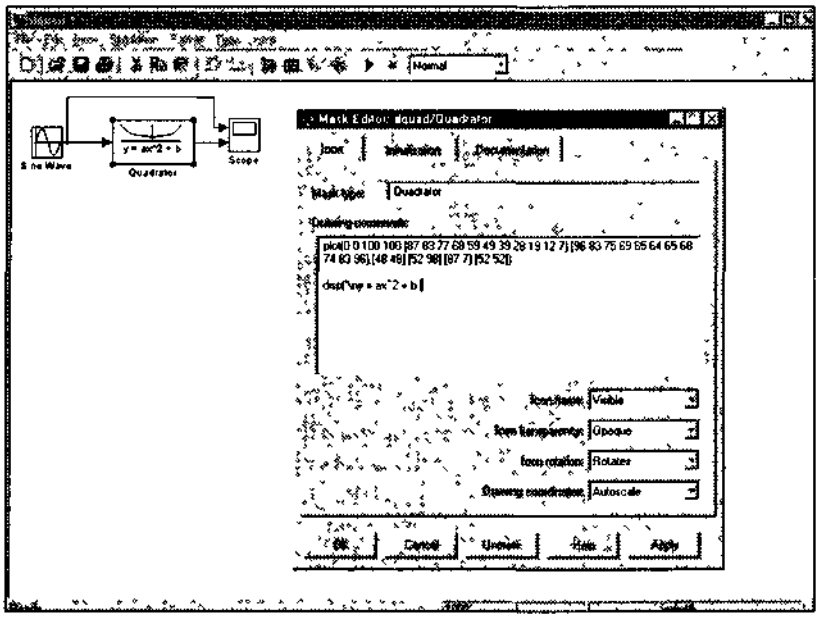


Рис. 10.13. Коррекция пиктограммы

Рисунок 10.14 иллюстрирует поворот созданной пиктограммы на 90 градусов с коррекцией связей между блоками и установленным па-

раметром Rotate. Как нетрудно заметить, на этот раз можно получить развернутую пиктограмму с правильным расположением надписей внутри нее.

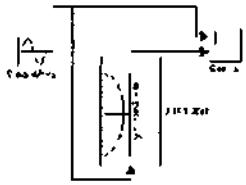


Рис. 10.14. Пример поворота пиктограммы, созданной с помощью редактора пиктограмм

Возможности создания красочных пиктограмм с помощью столь простого редактора, конечно, ограничены. Но выглядящие технически строго пиктограммы вполне можно создавать после небольшой практики работы с редактором.

Задание пиктограммы в виде готового рисунка

Пиктограмму в Simulink можно создать, используя практически любой графический редактор или файл графического формата, который поддерживается системой MATLAB. В число таких файлов входят файлы хорошо известных форматов PCX, JPG, TIF, BMP и др. Для загрузки такого файла служит графическая команда:

```
image(imread(имя_файла  тип_файла))
```

Пример создания пиктограммы на основе применения данной команды показан на рис. 10.15.

В данном случае в качестве рисунка для пиктограммы взята обложка книги (В. Дьяконов. MATLAB: учебный курс. СПб: Питер, 2001). Эту книгу можно рекомендовать тем читателям, которые хотят глубже изучить возможности базовой системы MATLAB, в частности возможности создания графических изображений для пиктограмм.

Создание библиотек пользователя

Требования к библиотекам пользователя

При серьезной работе с Simulink может потребоваться создание библиотек пользователя, составленных как из созданных им блоков, так

и блоков, взятых из встроенных в Simulink библиотек. Как показывает практика, большинству пользователей возможности встроенных библиотек Simulink кажутся явно избыточными и пользователи ощущают даже некоторый дискомфорт от постоянного поиска нужных им блоков.

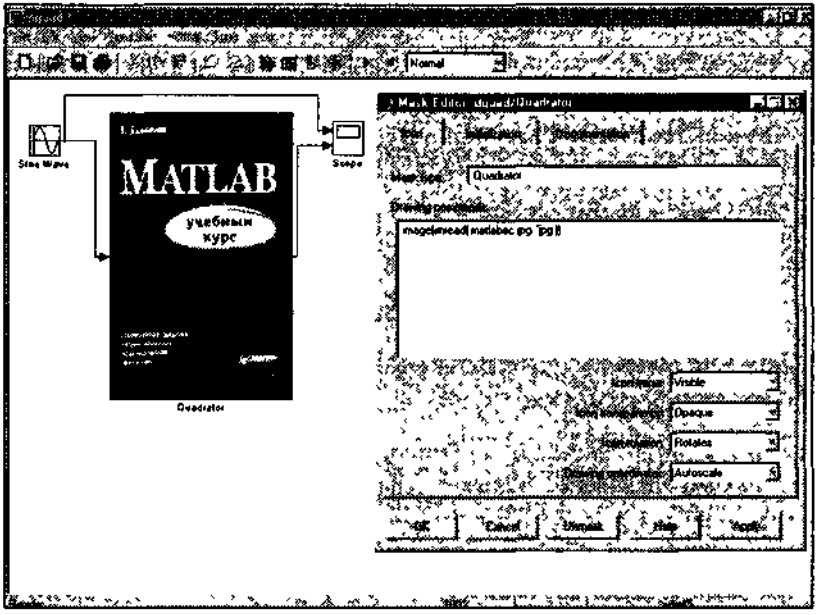


Рис. 10.15. Создание пиктограммы на основе графического файла

Выход из этой ситуации вполне очевиден — надо создавать свои новые библиотеки. Такие библиотеки по структуре и характеру применения должны удовлетворять правилам, существующим для встроенных библиотек:

- размещаться в своих окнах;
- иметь представление в виде пиктограмм блоков;
- иметь должную сопровождающую документацию.

При создании библиотек и их применении следует учитывать, что между блоком в модели и блоком в библиотеке устанавливается специальная связь. В прежних версиях MATLAB перенос блоков из какого-либо раздела встроенных библиотек в окно библиотеки пользователя требовал разрыва связи с блоками встроенных библиотек

и создания связи с блоками в окне библиотеки пользователя. Для этого применялась команда Break Library Link (Разорвать связь с библиотекой). В Simulink 4.0 эта команда исключена.

Окно библиотеки пользователя

Библиотека, или набор блоков пользователя, может быть создана в специальном окне библиотек пользователя. Это окно открывается командой File ▸ New ▸ Library (рис. 10.16).

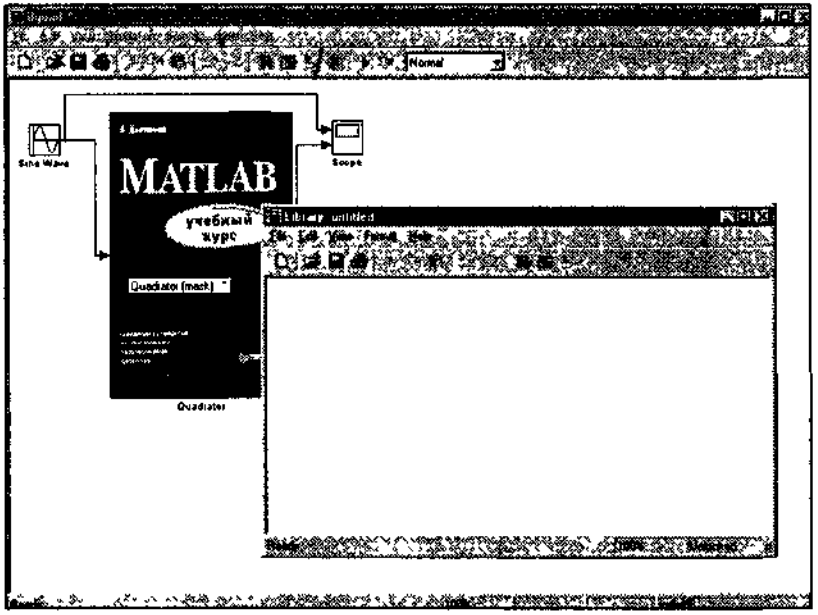


Рис. 10.16. Создание окна библиотеки пользователя

Нетрудно заметить, что это окно имеет несколько упрощенный интерфейс. В частности, в нем нет средств запуска процесса моделирования. Окно создается пустым. В конце строки состояния окна видно сообщение Unlocked, говорящее о том, что библиотека открыта и может изменяться и пополняться.

Перенос блоков в окно библиотеки

В Simulink 4.0 перенос библиотечных блоков в окно новой библиотеки пользователей заметно упростился. Достаточно, расположив

рядом окна браузера библиотек и новой библиотеки, перетащить в последнюю нужные блоки. Связи между ними и встроенной библиотекой редактировать не надо

Можно также перенести в окно новой библиотеки и созданные маскированные подсистемы. Прямо перетащить их в окно новой библиотеки нельзя — в ней действует «правило одностороннего движения». Это значит, что блоки можно перемещать из окна библиотеки в окно модели, но никак не наоборот.

Тем не менее есть путь выполнить и эту операцию. Для этого достаточно выделить нужную маску, командой Edit ▸ Copy меню окна моделей Simulink поместить маску в буфер, а затем (наметив курсором мыши положение) командой Edit ▸ Paste поместить маску в окно новой библиотеки. Рисунок 10.17 показывает созданную таким образом новую библиотеку mylib.

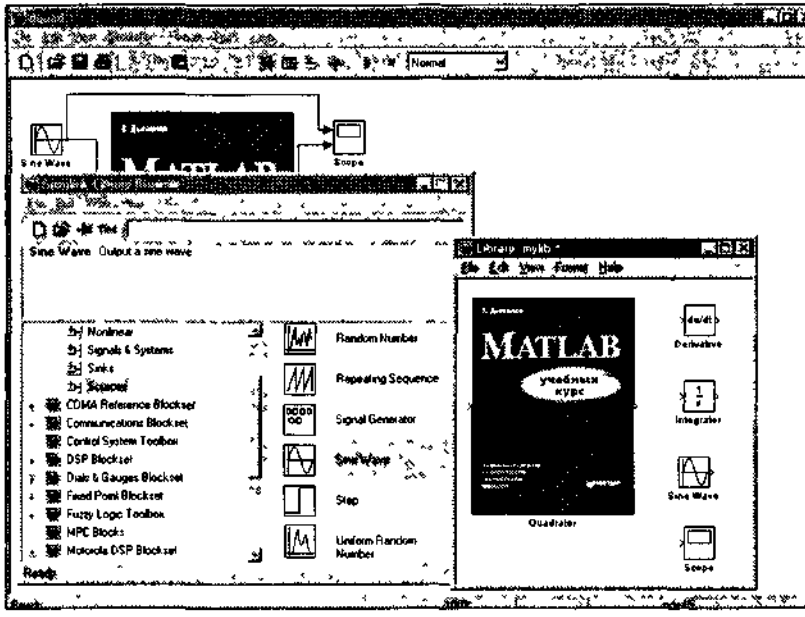


Рис. 10.17. *Ссозданная новая библиотека mylib*

После заполнения блоками новая библиотека блокируется командой меню Edit ▸ Locked. После этого библиотека становится недоступной для пополнения и модернизации. Впрочем, можно разбло-

кировать библиотеку командой Edit ▸ Unlocked и выполнить ее модернизацию

Завершается создание новой библиотеки ее записью на диск с помощью команды меню File ▸ Save As... окна новой библиотеки. Библиотека хранится в виде файла с заданным именем и расширением .mdl (таким же, как и у файлов моделей Simulink). Остается отметить, что внутри окна новой библиотеки можно создать окна ее разделов. Таким образом, структура библиотеки может быть многоуровневой.

Применение библиотек пользователя

Работа с библиотеками пользователя ничем не отличается от работы со встроенными в Simulink библиотеками. С помощью команды в виде имени библиотеки (ее файла) можно вызвать окно новой библиотеки. Из него, как обычно, мышью можно перетащить нужные блоки в создаваемую модель.

Глава 11. Инструментальные средства Simulink

- Меню Tools
- Работа с отладчиком графических S-моделей
- Браузер данных Simulink
- Установки просмотра
- Сравнение моделей
- Генератор отчетов Simulink
- Другие инструментальные средства

Меню Tools

Подготовка и запуск моделей в Simulink производятся настолько просто и наглядно, что большинству пользователей могут не пригодиться специальные отладочные средства. Обычно разработчик модели, используя метод проб и ошибок, постепенно совершенствует ее и добивается ее корректной работы. Для контроля состояния тех или иных блоков к ним подключаются регистраторы, например осциллографы или дисплеи, что позволяет оценивать уровни входных и выходных сигналов и их временные зависимости.

Тем не менее Simulink 4.0 имеет специальные инструментальные средства для отладки моделей. Они особенно важны, когда идет отладка моделей событийно управляемых систем, систем со сложными логическими алгоритмами и т. д. В этом случае немаловажную роль играет отладка модели по шагам с контролем активности блоков и некоторых параметров системы.

Основные инструментальные средства Simulink 4.0 сосредоточены в новом меню — Tools (рис. 11.1). Появление этого меню облегчает доступ к разнообразным инструментальным средствам Simulink 4.0, которые теперь собраны воедино.

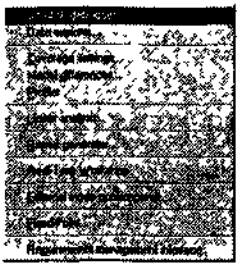


Рис. 11.1. Меню Tools окна моделей Simulink

Меню Tools имеет ряд команд:

- Simulink Debugger... — открытие отладчика моделей Simulink;
- Data Explorer... — открытие браузера данных;
- Coverage Setting... — открытие окна настроек отчета по моделированию;
- Model Differences... — открытие окна сравнения моделей;
- Profiler — открытие окна профилирования моделей;
- Linear analysis... — открытие окна линейного анализа;
- Report generator... — открытие генератора отчетов;
- Real Time Workshop — доступ к расширению Real Time Workshop;
- External mode control panel ... — вывод панели контроля моделирования при внешнем управлении;
- Fixed-point ... — доступ к расширению Fixed-point;
- Requirements management interface... — открытие окна просмотра модели.

Работа с отладчиком графических S-моделей

Запуск отладчика

Отладчик графических S-моделей обычно используется после загрузки или подготовки графической модели в окне моделей Simulink. Запустить отладчик можно тремя способами:

- при помощи пункта меню Tools ▸ Simulink Debugger... (в Simulink 4.0);

- нажатием кнопки Debug панели инструментов окна S-модели;
- выполнением команды sldebug в командной строке MATLAB:

```
sldebug(Имя_модели)
```

Рассмотрим графический интерфейс отладчика и его окно (рис. 11.2).

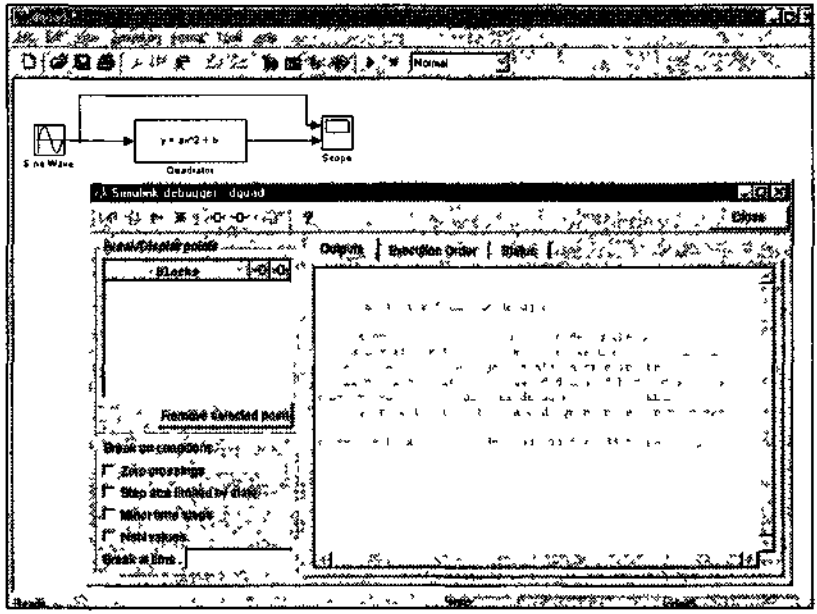


Рис. 11.2. Начало работы с отладчиком моделей

Панель инструментов отладчика

Окно отладчика не имеет меню — управление им происходит только с помощью кнопок панели инструментов (рис. 11.3).

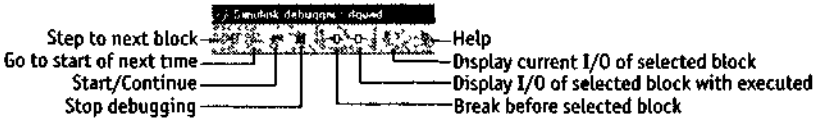


Рис. 11.3. Панель инструментов отладчика моделей

Эта панель имеет следующие кнопки:

- Step to next block — переход к следующему блоку;
- Go to start of next time — переход к следующему такту времени;

- Start/Continue — старт/продолжение отладки;
- Stop debugging — остановка отладки;
- Break before selected block — установка точки прерывания перед выделенным блоком;
- Display I/O of selected block with executed — показ исполняемых блоков ввода/вывода;
- Display current I/O of selected block — показ текущих блоков ввода/вывода;
- Help — вызов справки по отладчику.

Работа с отладчиком

Отладчик имеет множество различных режимов работы, из которых мы рассмотрим лишь основные. При запуске отладчика выводится окно с пустыми установками и открытой вкладкой Outputs (Выходы). На этой вкладке размещено короткое приглашение к работе с отладчиком и пояснение его назначения (рис. 11.2).

Работа с отладчиком основана на расстановке так называемых точек останова (break points) и точек показа (display points). Их список формируется в поле Blocks. Чтобы включить блок в список, надо выделить его и нажать кнопку Break before selected block — блок будет помещен в список Blocks. Для каждого блока можно задать опции останова и показа его выходного сигнала (установкой флажка после имени блока в списке).

При запуске модели будет происходить остановка моделирования перед каждым из блоков, включенных в список Blocks, с выводом результатов на каждом шаге. При этом блок, на котором произошла остановка моделирования, выделяется цветом фона. Для удаления выделенных блоков из списка Blocks служит кнопка Remove select points.

На рис. 11.4 дан пример пошаговой работы модели, в котором контролируются блоки источника синусоидального сигнала Sine Wave, умножителя Product и сумматора Sum. Для выполнения очередного шага отладки следует нажать кнопку Start/Continue панели инструментов.

Как видно из рис. 11.4, на вкладке Outputs отражается текущее модельное время Tm, имя контролируемого блока (в виде индекса блока @s:b, где s — номер S-модели и b — номер блока), значение его входного параметра U, и выходного Y. Их анализ позволяет оценить корректность работы блока.

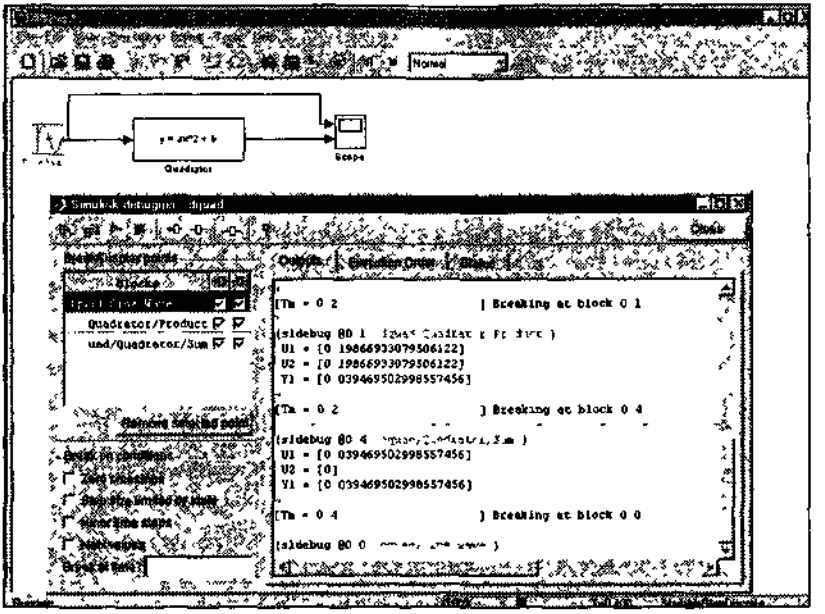


Рис. 11.4. Момент тестирования блока Sine Wave

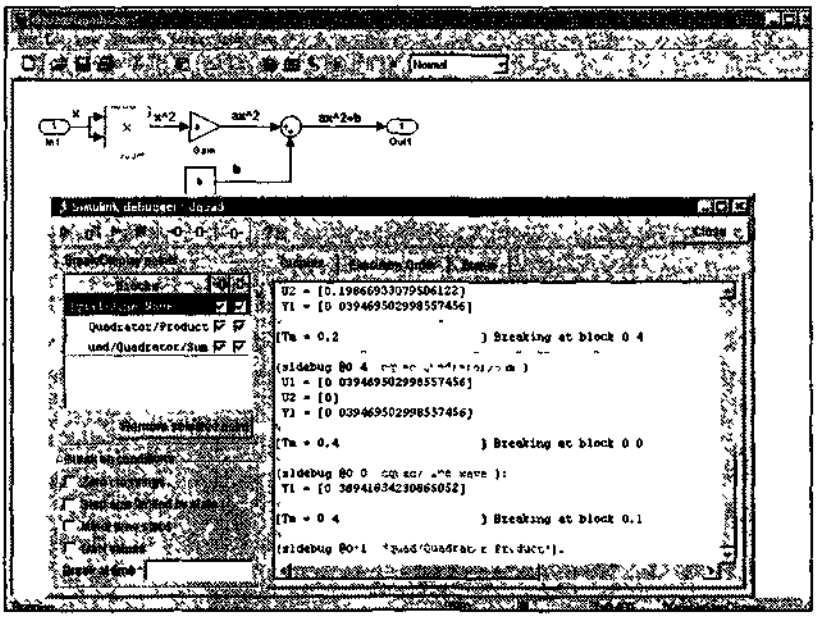


Рис. 11.5. Тестирование блока Product в маскированной подсистеме

Рисунок 11.4 представляет момент отладки, когда останов произошел на блоке Sine Wave. В ходе отладки блоки из списка тестируются последовательно. Если в данный момент тестируется блок, расположенный в подсистеме (в том числе маскированной), то открывается окно с ее графической моделью, в котором выделен тестируемый блок. Этот случай иллюстрирует рис. 11.5, на котором выделен блок умножителя Product, расположенный в маскированной подсистеме Quadrator.

Помимо кнопки последовательной пошаговой отладки по списку Blocks можно воспользоваться двумя первыми кнопками в панели инструментов. Кнопка Step to next block обеспечивает отладку по шагам с переходом к следующему блоку (то есть от одного выделенного блока к другому). А кнопка Go to start of next time дает переход к следующему такту времени.

Дополнительные возможности отладчика

Отладчик позволяет установить различные типы точек останова, например безусловные и условные. Раздел окна отладчика Break on condition задает опции, обеспечивающие останов при выполнении различных условий:

- Zero crossing — прохождение сигнала через нулевой уровень;
- Step size limited by state — превышение допустимого значения шага;
- Minor time steps — недопустимо малый шаг времени;
- NaN values — появление нечисленного значения (NaN);
- Break at time — остановка в заданный момент времени.

Проверка порядка выполнения блоков

Вкладка Execution Order окна отладчика S-моделей позволяет получить информацию о порядке выполнения блоков в ходе моделирования (рис. 11.6).

Содержание этой вкладки очевидно. Обратите внимание лишь на то, что в маскированных подсистемах отражаются все входящие в них блоки.

Оценка состояния отладчика

Состояние отладчика на текущем шаге моделирования можно оценить по информации, приводимой на вкладке Status (рис. 11.7).

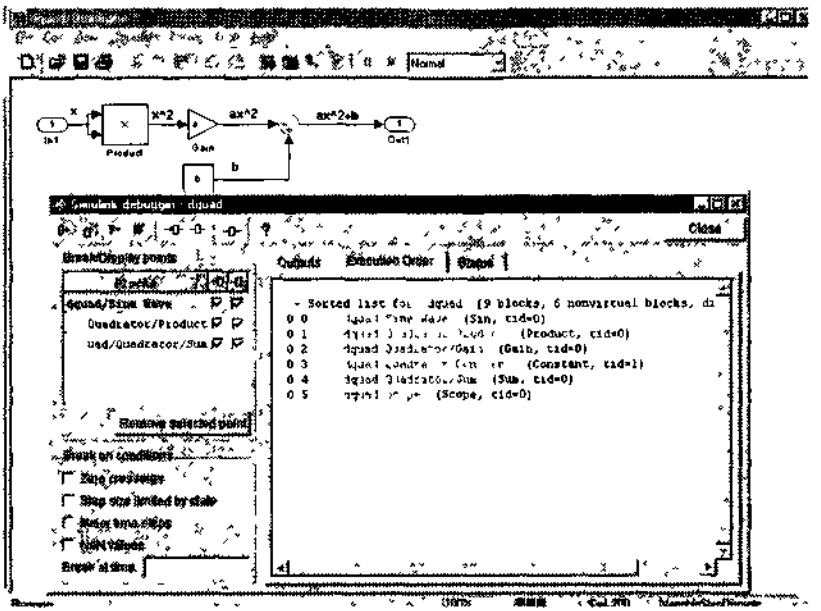


Рис. 11.6. Просмотр порядка выполнения блоков

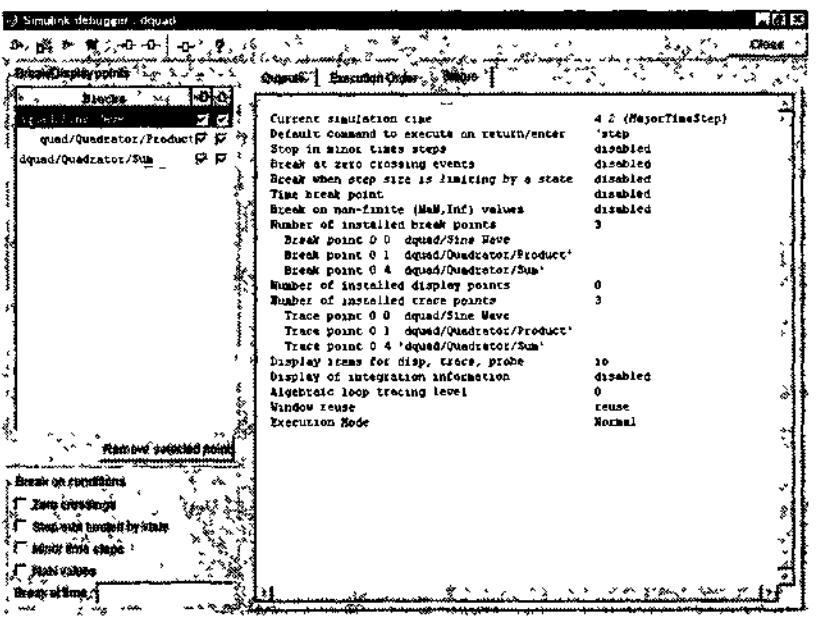


Рис. 11.7. Оценка состояния отладчика

Как нетрудно заметить, это состояние оценивается значением текущего времени моделирования, используемой по умолчанию командой исполнения отладки, степенью активности команд останова (stop) и прерывания (break), числом и характером точек прерывания и трассировки и др. Всего используется около полутора десятков признаков состояния, и все они видны на упомянутой вкладке.

Управление отладчиком из командной строки MATLAB

С появлением в Simulink 4.0 отладчика с удобным графическим интерфейсом целесообразность работы с отладчиком из командной строки MATLAB резко уменьшилась. Тем не менее возможность отладки из командной строки сохранена. Ниже приводится список команд отладчика, доступных из командной строки MATLAB. Обратите внимание, что некоторые команды допускают повтор (Yes), другие нет (No).

Команда	Краткая форма	Повтор	Описание
ashow	as	No	Показ зацикливания
atrace	at	No	Установка циклов трассировки
bafter	ba	No	Вставка точки останова при выходе из блока
break	b	No	Вставка точки останова при входе в блок
bshow	bs	No	Показ указанного индексом блока
clear	cl	No	Удаление точки останова
continue	c	Yes	Продолжение моделирования
disp	d	Yes	Показ I/O блоков при останове
help	?orh	No	Вывод справки по командам отладчика
ishow	i	No	Включение/выключение вывода общей информации о моделировании
minor	m	No	Включение/выключение режима minorstepmode — отладки с использованием второстепенных шагов
nanbreak	na	No	Установка/удаление остановок при небесконечных значениях
next	n	Yes	Переход к состоянию в следующий момент времени
probe	p	No	Показ сигналов на входе/выходе блоков

продолжение

Команда	Краткая форма	Повтор	Описание
quit	q	No	Прерывание моделирования
run	r	No	Запуск моделирования
slist	sli	No	Список невиртуальных блоков
states	state	No	Показ значений текущего состояния
status	stat	No	Показ параметров отладчика
step	s	Yes	Переход к следующему блоку
stop	sto	No	Остановка моделирования
systems	sys	No	Список моделей невиртуальных систем
tbreak	tb	No	Установка/стирание остановок по времени
trace	tr	Yes	Показ входов/выходов блоков во время выполнения
undisp	und	Yes	Удаление блока из списка отображаемых
untrace	unt	Yes	Удаление блока из списка трассировки
xbreak	x	No	Прерывание при переменном шаге времени или состоянии, требующем ограничения шага
zcbreak	zcb	No	Прерывание при обнаружении непредусмотренного пересечения нуля
zclist	zcl	No	Список блоков, дающих непредусмотренное пересечение нуля

Пример работы с отладчиком в командном режиме представлен на рис. 11.8. При управлении из командной строки отладчик дает ту же информацию, что и при управлении из окна отладчика.

Для более детального знакомства с командами отладчика, используемыми в командном режиме MATLAB, можно использовать команду h или ?, выводящую полный список команд отладчика с их опциями.

ВНИМАНИЕ

Работу с отладчиком в командном режиме MATLAB можно рекомендовать только достаточно опытным пользователям. Этот режим предоставляет наиболее полный доступ к средствам управления моделями — например, позволяет просматривать результаты промежуточных вычислений и выполнять во время останова операции системы MATLAB.

Браузер данных Simulink

Для контроля данных, которые используются пакетом Simulink, служит браузер данных. Его окно (рис. 11.9) вызывается при помощи команды меню Tools ▸ Data Explorer...

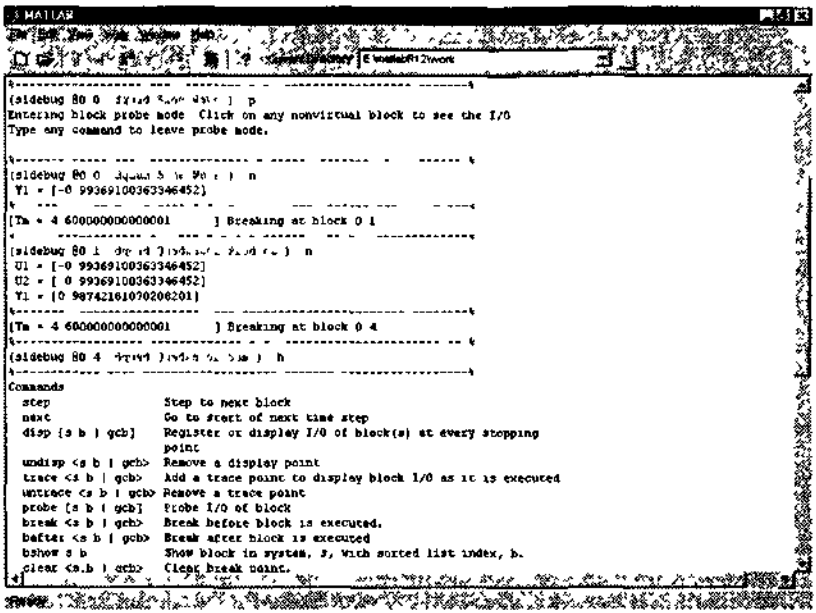


Рис. 11.8. Работа с отладчиком S-моделей из командной строки MATLAB

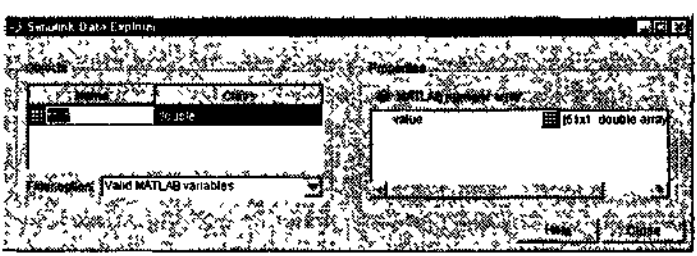


Рис. 11.9. Окно браузера данных Simulink

Окно браузера данных имеет два раздела. В левом имеется список использованных объектов с указанием их типа (Class). В правом разделе (Properties) указывается подробная характеристика выделенного объекта.

Настройка отчета

Simulink обеспечивает детальный анализ данных моделирования с помощью специального отчета. Команда меню Tools ▸ Coverage Setting... выводит окно с параметрами отчета (рис. 11.10).

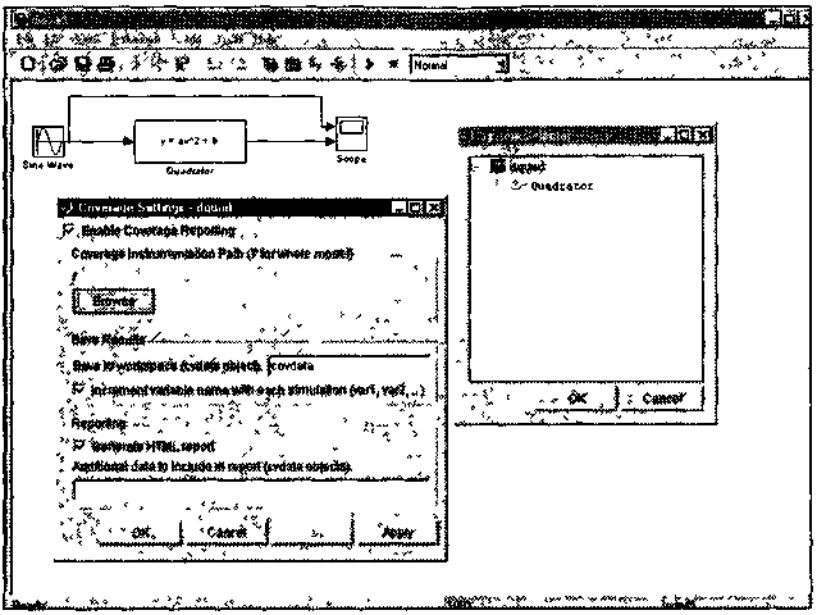


Рис. 11.10. Окно настройки отчетов

Это окно по умолчанию содержит обычно сброшенный флажок Enable Coverage Reporting (Включить отчетность). Если установить этот флажок, то станет доступным ряд установок. Так, нажатием кнопки Browse можно открыть окно поиска пути к модели и ее подсистемам, по моделированию которых составляется отчет (рис. 11.10). Можно также задать запись объектов в рабочее пространство, автоматическое увеличение индекса в именах переменных и задание генерации отчетов в виде HTML-файлов.

Сравнение моделей

В ходе создания нужной модели пользователь нередко создает множество моделей с различными значениями параметров. Иногда бывает трудно выяснить, чем разные варианты моделей отличаются друг от друга. В этом случае помогает специальная команда меню Tools ▸ Model differences... Открывающееся при этом окно Model Differencing Tool представлено на рис. 11.11.

Это окно вначале пусто. Для сравнения двух моделей нужно загрузить их в каждое из окон двойного окна с помощью кнопок с изображением

открывающейся папки. При этом появляются отдельные окна сравниваемых моделей, показанные на рис. 11.11 справа от двойного окна. После этого можно перемещаться по дереву блоков моделей и выполнять их сравнение. Если будут обнаружены блоки с одинаковыми именами и разными значениями параметров, то сообщение об этом появится внизу окна.

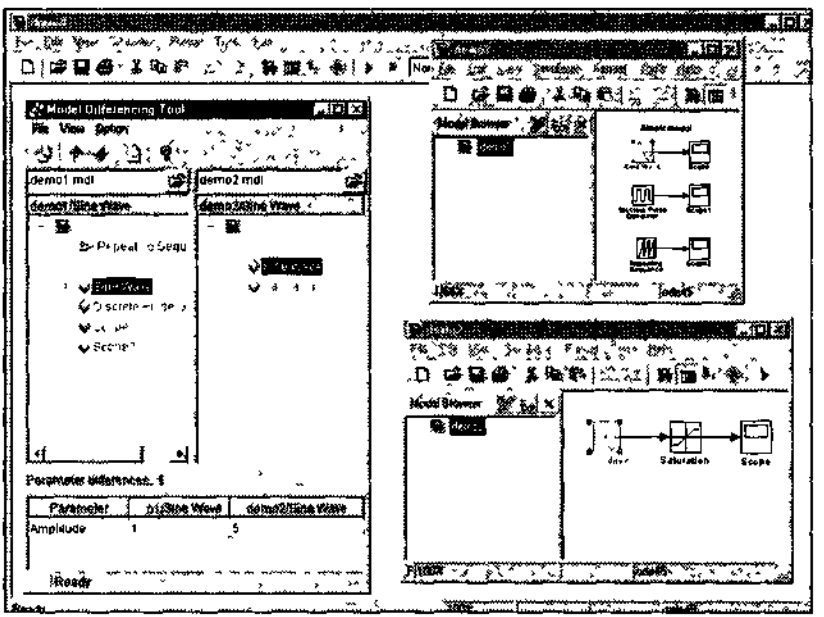


Рис. 11.11. Сравнение двух моделей

Генератор отчетов Simulink

Что такое отчет

Перед любым разработчиком модели устройства после создания и отладки модели возникает весьма специфическая обязанность — подготовить отчет. Отчет должен содержать детальное описание модели, ее блоков, значений параметров и т. д., вплоть до приведения полученных при моделировании осциллограмм.

В новых реализациях MATLAB и Simulink появились специальные средства, блестяще решающие эту задачу, — генераторы отче-

тов. В Simulink 4.0 генератор отчетов доведен до высокой степени совершенства — по полноте отчета и красочности его подготовки с Simulink трудно сравниться даже хорошему инженеру с качествами отличного оформителя.

Запуск генератора отчетов

Для запуска генератора отчетов надо выполнить команду меню Tools ▸ Report generator... Появится окно генератора отчетов, представленное на рис. 11.12.

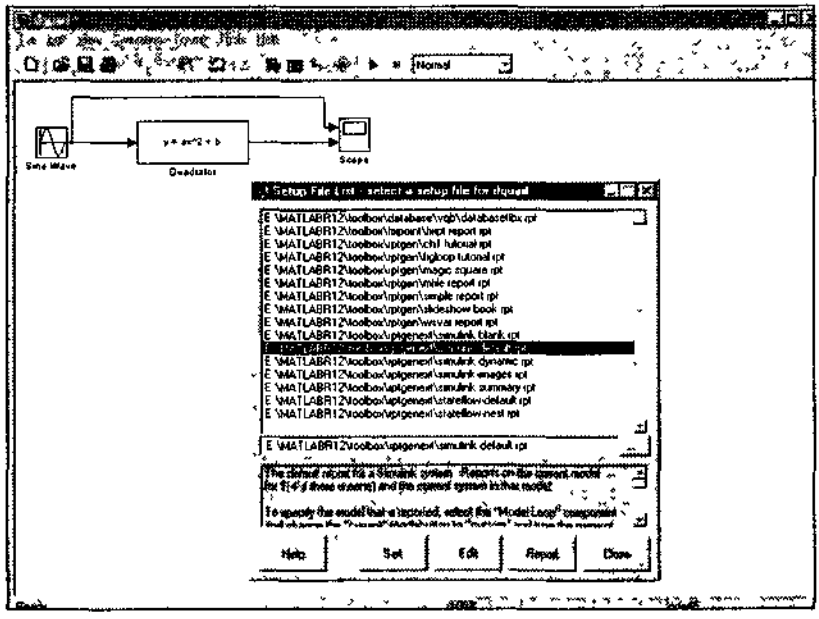


Рис. 11.12. Окно генератора отчетов

В этом окне имеется ряд файлов-шаблонов для подготовки отчетов. Чтобы формировать отчет по шаблону, выберите требуемый шаблон и нажмите кнопку Set.

Редактирование отчета

Генератор отчетов позволяет выполнить детальное редактирование отчета. Для этого нужно нажать кнопку Edit в окне генератора отчетов.

Однако надо отметить, что редактирование отчетов — процесс отнюдь не простой. Об этом свидетельствует окно редактирования, представленное на рис. 11.13.

ВНИМАНИЕ

Генератор отчетов — сложная программа, и ее полное описание выходит за рамки данной книги Пользователям, не слишком искушенным в применении пакета Simulink, можно порекомендовать работать с установками генератора отчетов, принятыми по умолчанию

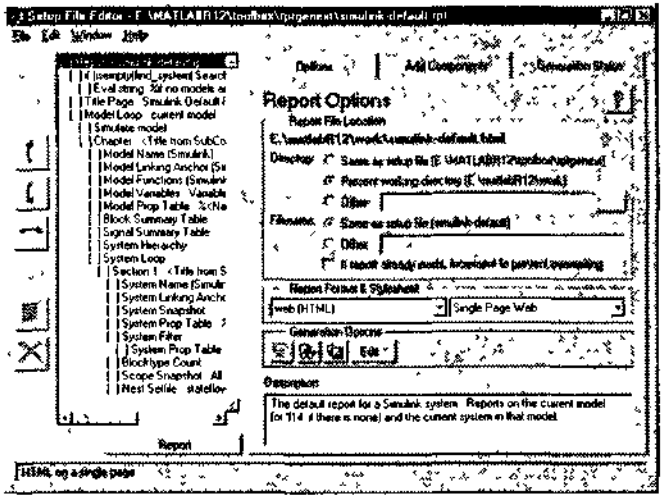


Рис. 11.13. Окно редактирования отчетов

Пример подготовки отчета

Для подготовки отчета по текущей модели достаточно нажать кнопку Report окна генератора отчетов. Генератор автоматически выполнит моделирование схемы и выведет окно Интернет-браузера с подготовленным отчетом Заметим, что для подготовки отчета в формате HTML нужно в окне Coverage Settings (см. рис. 11.10) установить флажок Generate HTML report.

На рис. 11 14 показано начало отчета в окне браузера Internet Explorer 5.0.

Уже из начала отчета видно, что он составлен в лучших традициях HTML-документов Каждый раздел отчета представлен гиперссыл-

кой, активизация которой переводит соответствующий раздел в область просмотра браузера Просмотрим созданный отчет, чтобы получить представление о его структуре и содержании Отметим, что данный отчет выполнен по стандартной форме (по умолчанию) и генератор отчетов Simulink имеет множество других форм

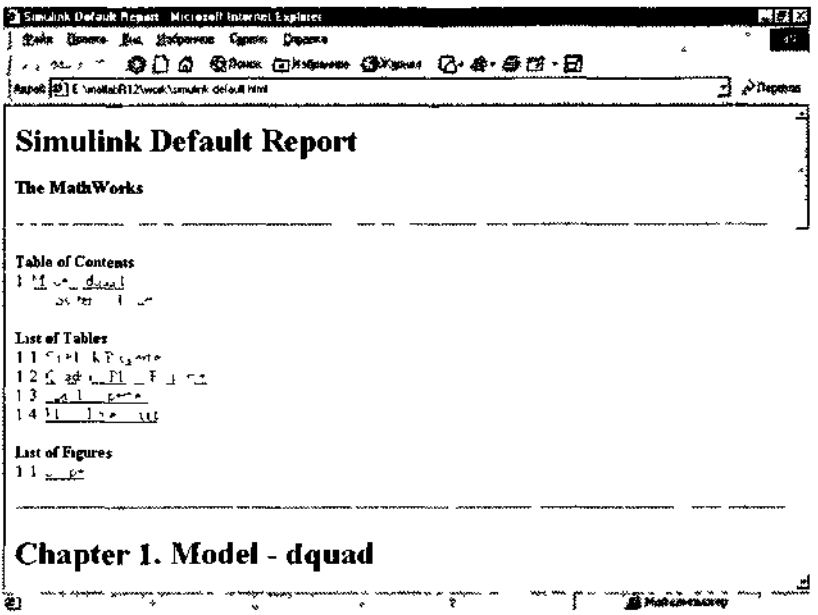


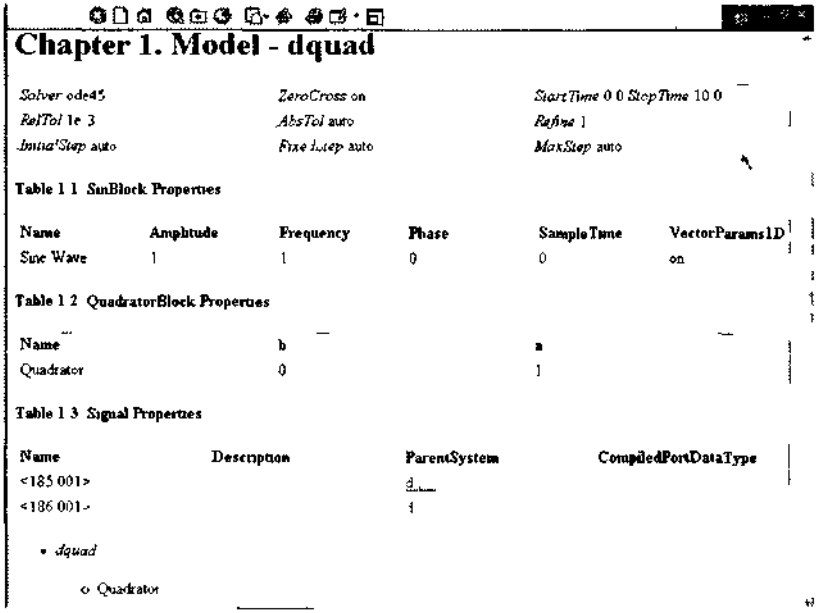
Рис. 11.14. Начало HTML-отчета по модели dquad

Рисунок 11 15 показывает раздел отчета с описанием модели Он начинается с указания того, какой решатель дифференциальных уравнений использован и какие параметры были для него заданы Затем идут подробные таблицы с описанием блоков модели

На рис 11 16 показано продолжение описания модели Здесь приводится графическое изображение модели и спецификация ее блоков

Наконец, на рис 11 17 представлена заключительная часть отчета — осциллограммы модели

Итак, полученный отчет отличается детальностью Его можно использовать как непосредственно в отчетных материалах по работе, так и для подготовки справочных материалов для библиотек пользователя моделей



Chapter 1. Model - dquad

Solver ode45 *ZeroCross* on *StartTime* 0 0 *StopTime* 10 0
RelTol 1e-3 *AbsTol* auto *Refine* 1
InitialStep auto *FixedStep* auto *MaxStep* auto

Table 1-1 SinBlock Properties

Name	Amplitude	Frequency	Phase	SampleTime	VectorParams1D
Sine Wave	1	1	0	0	on

Table 1-2 QuadratorBlock Properties

Name	b	a
Quadrator	0	1

Table 1-3 Signal Properties

Name	Description	ParentSystem	CompiledPortDataType
<185:001>		dquad	
<186:001>		dquad	

- *dquad*
  - Quadrator

Рис. 11.15. Описание модели и ее блоков в отчете

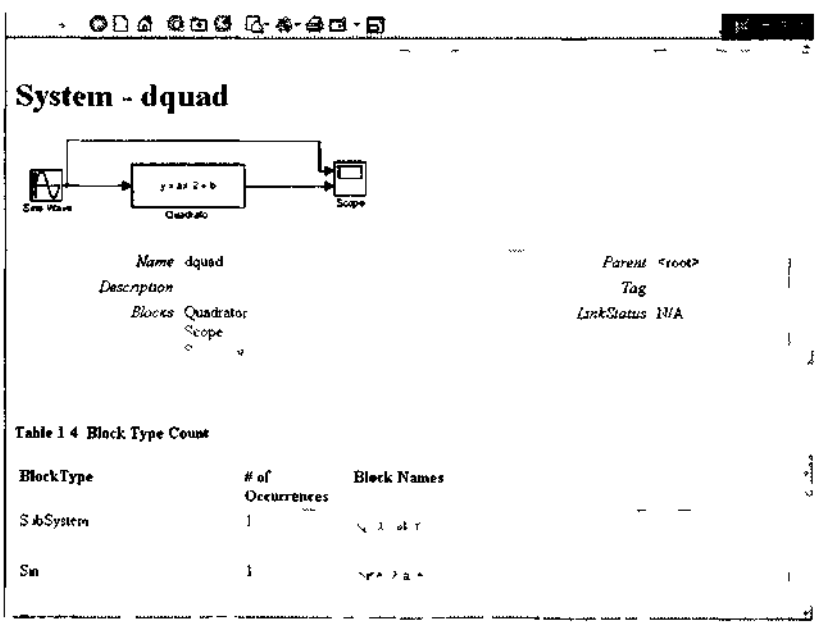


Рис. 11.16. Изображение модели и спецификация блоков в отчете

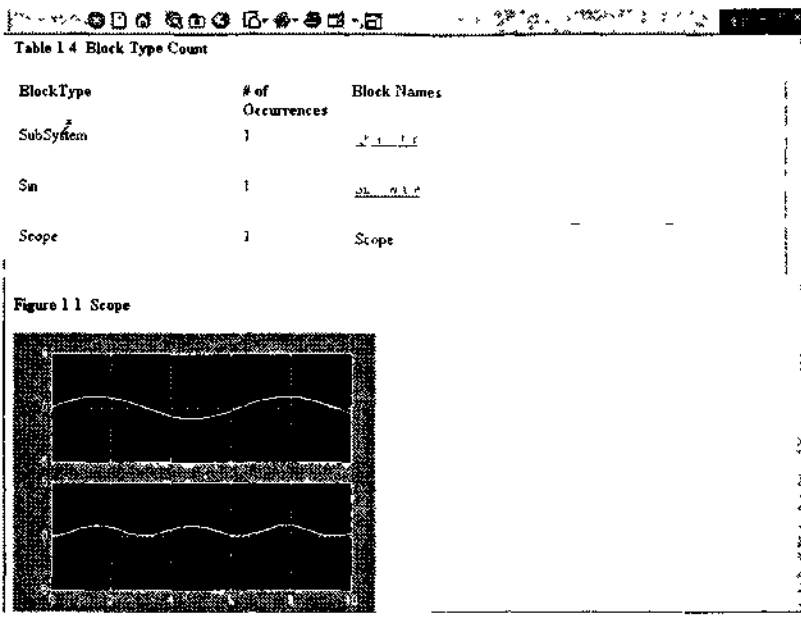


Рис. 11.17. Осциллограммы модели в отчете

Другие инструментальные средства

В меню Tools имеется ряд других специальных инструментальных средств. Отметим их коротко.

- Команда Profiler включения/выключения профилирование моделей. Это средство, обеспечивающее оценку скорости выполнения того или иного шага моделирования. По умолчанию эта опция выключена.
- Команда Linear analysis... служит для открытия окна с результатами линейного анализа систем. Эта команда полезна для специальных моделей, в которых реализован этот вид анализа.
- Команда Real-Time Workshop открывает доступ к так называемой мастерской реального времени, которая описана в главе 16. Пока отметим, что эта мастерская позволяет управлять процессом моделирования Simulink с помощью внешних устройств, которыми может оснащаться компьютер. В результате становится возможной организация компьютеризованных систем, работающих

в реальном масштабе времени. Команда External mode control panel открывает окно с установками для работы Simulink в режиме внешнего управления

- Команда Fixed-point открывает окно для работы с расширением Simulink Fixed-point
- Команда Requirements managements interface... открывает окно интерфейсного навигатора Requirements managements interface Navigator, предс гавленное на рис. 11.18.

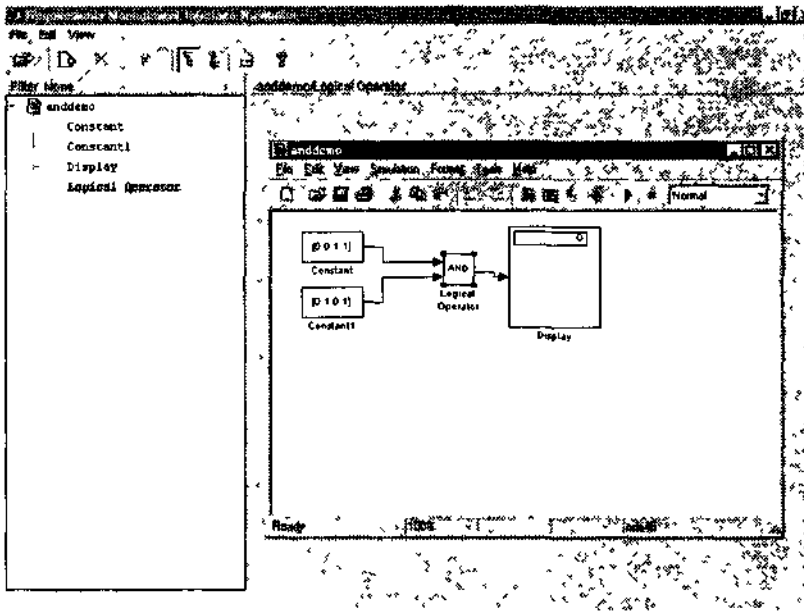


Рис. 11.18. Окно интерфейса

Загрузив модель, в левой части этого окна можно наблюдать дерево ее блоков. При выделении блока появляется окно с моделью, в котором указанный блок будет выделен. Разумеется, это средство целесообразно применять при анализе достаточно сложных моделей.

Глава 12. Пакет Nonlinear Control Design

- Пакет Nonlinear Control Design (NCD) Blockset 5.0
- Примеры моделирования и оптимизации
- Особенности решаемых задач

Пакет Nonlinear Control Design (NCD) Blockset

Назначение пакета NCD

Новая версия пакета прикладных программ для построения нелинейных систем управления Nonlinear Control Design (NCD) Blockset 5.0 служит для динамической оптимизации систем управления. Он обеспечивает:

- легкую настройку переменных;
- указание неопределенных параметров систем,
- интерактивную оптимизацию;
- моделирование методом Монте-Карло;
- поддержку проектирования как одномерных, так и многомерных систем управления;
- моделирование подавления помех;
- моделирование процессов слежения;
- моделирование объектов с запаздыванием;
- решение других задач управления.

Пакет входит в число четырех основных пакетов расширения Simulink, объединенных под названием Blockset (см. рис. 1.1).

Состав блоков пакета NCD

Рисунок 12.1 показывает окно браузера библиотек, демонстрирующее доступ к блокам пакета NCD. Обозначение RMS относится к

понятию «root mean square» (корень квадратный из среднеквадратического значения).

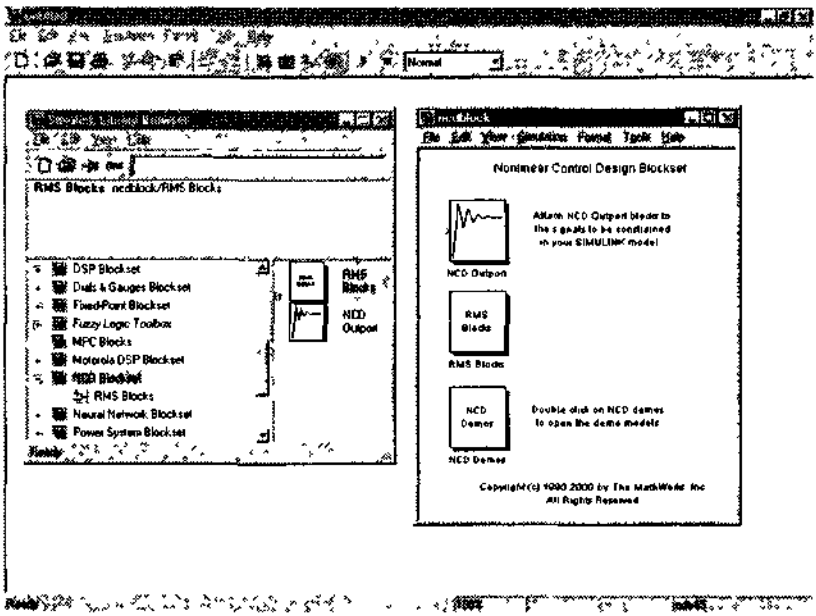


Рис. 12.1. Состав NCD Blockset

Набор блоков пакета NCD содержит всего три блока: CRMS (Continue RMS), DRMS (Discrete RMS) и NCD Output.

Блок CRMS реализует математическую зависимость

$$y(t) = \sqrt{\frac{1}{t}\int_0^t u^2(\tau)d\tau} \quad \text{при } t > 0,$$

где $u(t)$ — входной сигнал блока, $y(t)$ — его выходной сигнал. Следует отметить, что для стационарных случайных процессов с нулевым математическим ожиданием выходной сигнал блока при $t \rightarrow$ является среднеквадратическим отклонением.

Блок DRMS, по сути, реализует ту же зависимость, что и блок CRMS, но для сигналов, определенных в дискретные моменты времени:

$$y(n) = \sqrt{\frac{1}{n}\sum_{k=0}^{n} u^2(k)} \quad \text{при } n > 0.$$

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 выполнить следующие операции:

- задать требуемые ограничения во временной области на любой сигнал оптимизируемой системы;
- указать параметры, подлежащие оптимизации,
- указать неопределенные параметры;
- провести параметрическую оптимизацию системы с учетом заданных ограничений.

Работа с данным блоком будет рассмотрена подробно в дальнейшем.

Демонстрация работы блоков пакета NCD

Для запуска демонстрационного примера в режиме командной строки MATLAB введем команду

```
» rmsdemo
```

В результате выполнения этой команды появится рабочее окно Simulink с моделью, содержащей источник синусоидального сигнала с единичной амплитудой, к выходу которого подключены блоки CRMS и DRMS (рис. 12.2). С помощью блока Mux их выходы подключаются ко входу виртуального осциллографа.

Активизируем блок Scope (двойным щелчком мыши) и запустим процесс моделирования. Его результат отображен на осциллограммах рис. 12.2, где сплошная линия — выход блока CRMS, а ступенчатая — выход блока DRMS. Они также отличаются цветом линий.

С течением времени сигналы на выходах обоих блоков стремятся к одному и тому же установившемуся значению — действующему значению синусоиды с единичной амплитудой, равному $1/\sqrt{2} \approx 0.707$. Для блока CMRS это приближение получается плавным, а в случае блока DRMS — дискретным.

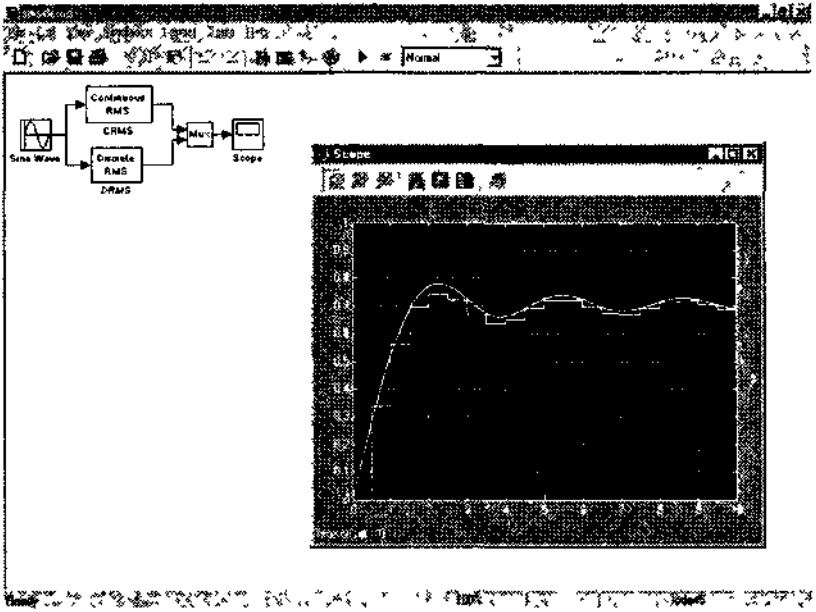


Рис. 12.2 Модель, иллюстрирующая функционирование блоков CRMS и DRMS

Примеры моделирования и оптимизации

Оптимизация коэффициента передачи И-регулятора

Рассмотрим пример параметрической оптимизации замкнутой системы автоматического регулирования. Для этого в режиме командной строки выполним команду

```
» ncdtut1
```

Это приведет к открытию окна Simulink с моделью исследуемой системы, представленной на рис. 12.3

Данная система представляет собой замкнутую структуру, состоящую из следующих компонентов:

1. Объект регулирования с передаточной функцией

$$\frac{\omega_0^2 e^{-p\tau}}{p^2 + 2\omega_0 \xi p + \omega_0^2}$$

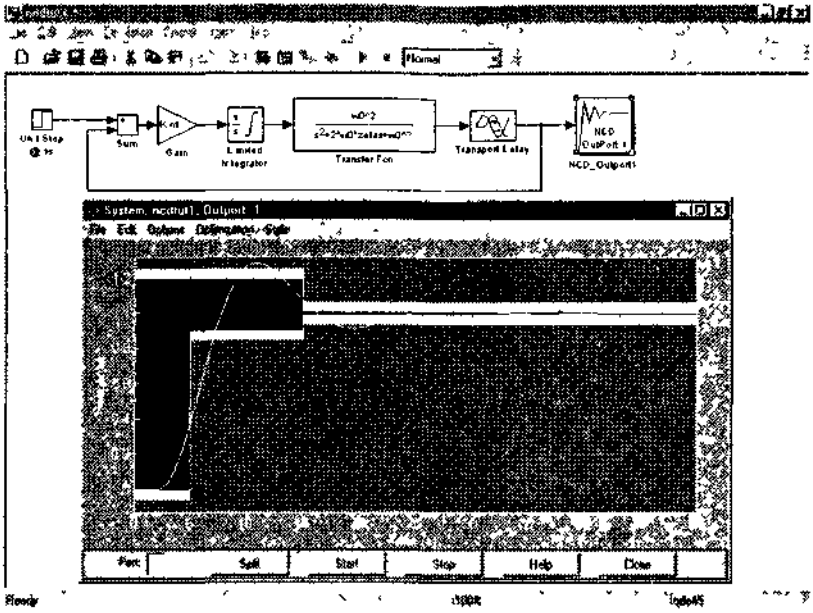


Рис. 12.3. Модель системы И-регулятора

который в модели отображен звеном второго порядка (блок Transfer Fcn) и звеном задержки (блок Transport Delay) со значениями $t = 1$ с, $\omega_0 = 1$ с$^{-1}$, $x = 1$ и обозначениями w0 = ω_0, zetas = x.

2. Интегральный регулятор (И-регулятор), в состав которого входят последовательно соединенные пропорциональное звено с коэффициентом передачи Kint и интегрирующее звено с ограничением выходного сигнала (блок Limited Integrator).
3. Контур обратной связи и звено сравнения Sum.

В модель также введены источник входного сигнала в виде единичного скачка и NCD-блок NCD Output, подключенный к выходу системы (блок с именем NCD OutPort 1)

Нетрудно видеть, что в данном случае контролируемым сигналом является реакция системы на единичный скачок, то есть ее переходная характеристика. 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. описание блока NCD Output выше) уже выполнены. Для выполнения третьего этапа в командной строке MATLAB наберем:

```
>> zeta=1
>> w0=1
>> Kint=0.3
```

Далее двойным щелчком мыши на блоке NCD OutPort 1 откроем его рабочее окно (рис. 12.3).

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 в положение, показанное на рис. 12.4, соответствующее нашим требованиям.

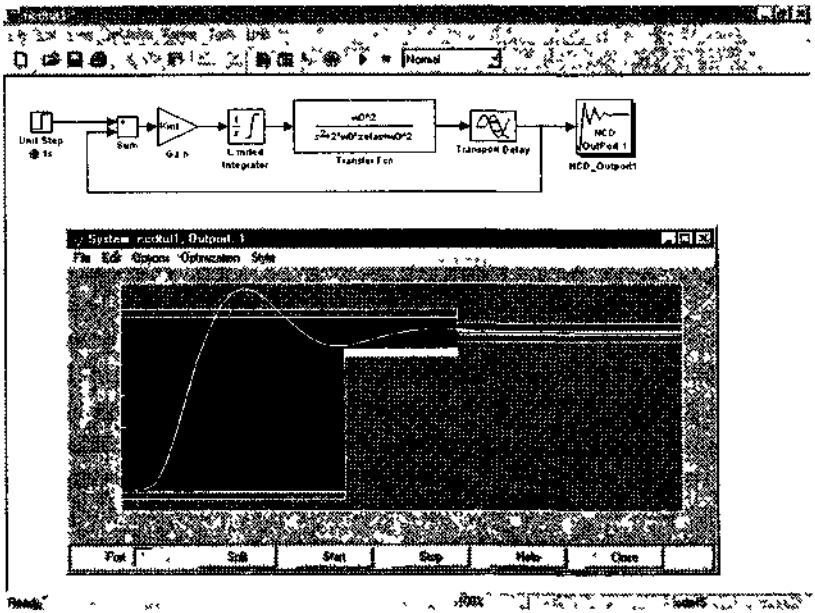


Рис. 12.4. Временные ограничения, соответствующие условиям задачи

Чтобы провести точную установку линий ограничения, следует выбрать требуемые линии с помощью щелчка мышью (выбранная линия становится белой) и воспользоваться пунктом меню Edit ▸

Edit constraint... В появившемся окне редактора ограничений (Constraint Editor) в строке Position Editor задайте начальную и конечную точки прямой в формате [x1 y1 x2 y2], а затем нажмите кнопку Done (рис. 12.5).

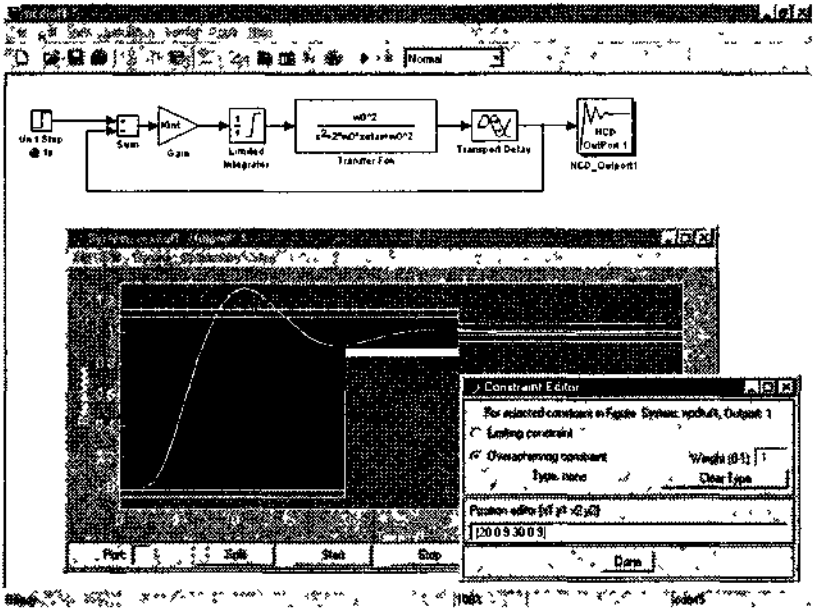


Рис. 12.5. Окно редактора ограничений

Точно такие же действия можно обеспечить, не используя пункты меню, а просто щелкнув правой кнопкой мыши на линии, что вызовет появление окна редактора ограничений.

Следующий этап — указание переменных, подлежащих оптимизации. Выбор пункта меню Optimization/Parameters... приведет к открытию диалогового окна Optimization Parameters, в котором задаются значения параметров оптимизации и интервал дискретизации (см. рис. 12.6).

В поле 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.

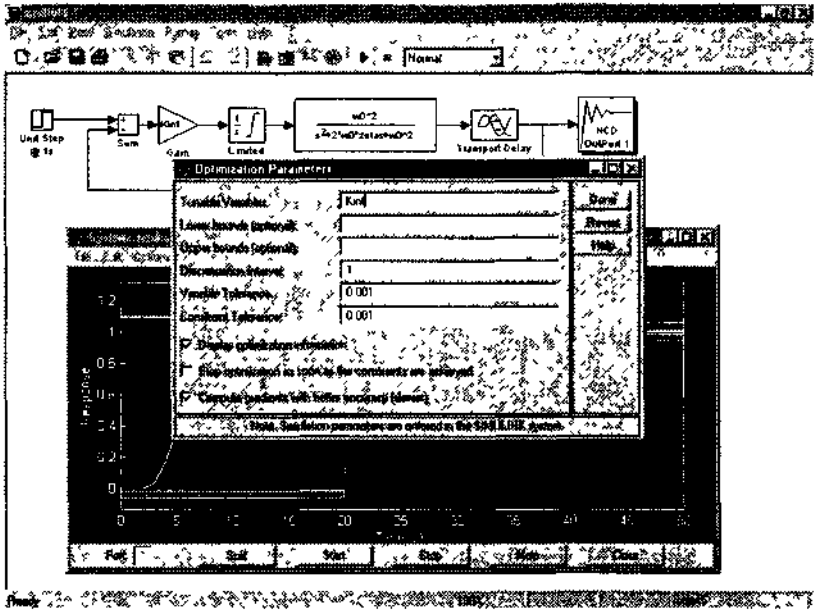


Рис. 12.6. Настройка параметров оптимизаци и интервала дискретизации

Теперь все готово для решения задачи. Процесс поиска решения можно запустить нажатием кнопки Start в панели инструментов окна модели Simulink. Поиск иллюстрируется начальной и конечной формами переходного процесса, представленными на рис. 12.7. В окне MATLAB также выводятся данные о ходе оптимизации.

Из рис. 12.7 видно, что оптимизированная переходная характеристика полностью соответствует заданным ограничениям.

Найденную оптимальную величину параметра Kint можно получить, набрав в командной строке имя данного параметра, то есть:

```
>> Kint
Kint =
    0.1961
```

Задачу оптимизации можно усложнить, введя неопределенные параметры. К таким параметрам обычно относят какие-либо 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 пунктом меню Optimization ▸ Uncertainty... окна NCD Output. Это приведет к открытию диалогового окна Uncertain Variables, в котором задаются неопределенные переменные. Введем указанные значения согласно рис. 12.8.

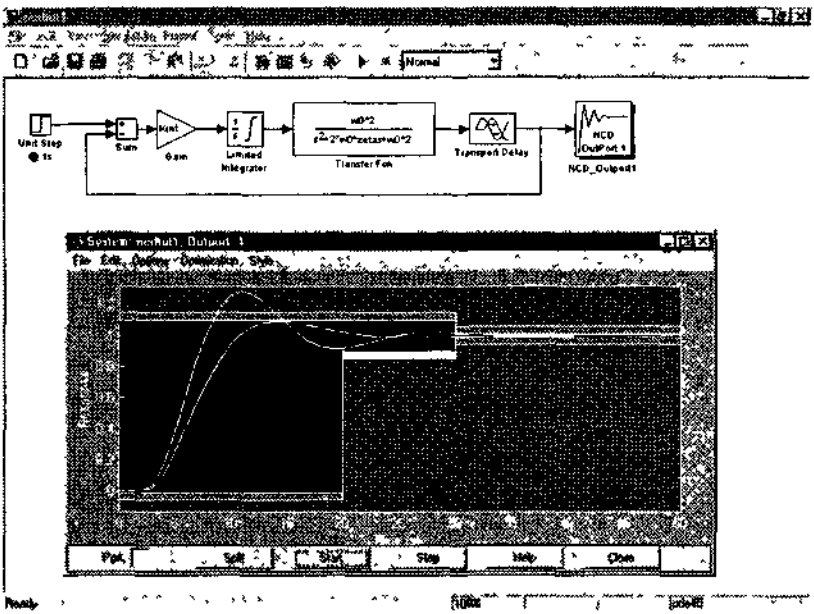


Рис. 12.7. Иллюстрация процесса оптимизации

Заметим, что по умолчанию используется номинальное значение параметра (Constrain nominal simulation). Для введения неопределенности необходимо задать нижнюю (Constrain lower simulation) и/или верхнюю (Constrain upper simulation) границы диапазона неопределенности. Установка флажка Constrain Monte Carlo simulations позволяет провести моделирование для нескольких случайных значений указанных параметров внутри отмеченной зоны (метод Монте-Карло). Число таких значений задается в поле Number of Monte Carlo simulations; не рекомендуется выбирать его очень большим ввиду возможного значительного увеличения времени счета

Завершив задание неопределенных переменных нажатием кнопки Done, можно продолжить оптимизацию системы в указанных условиях.

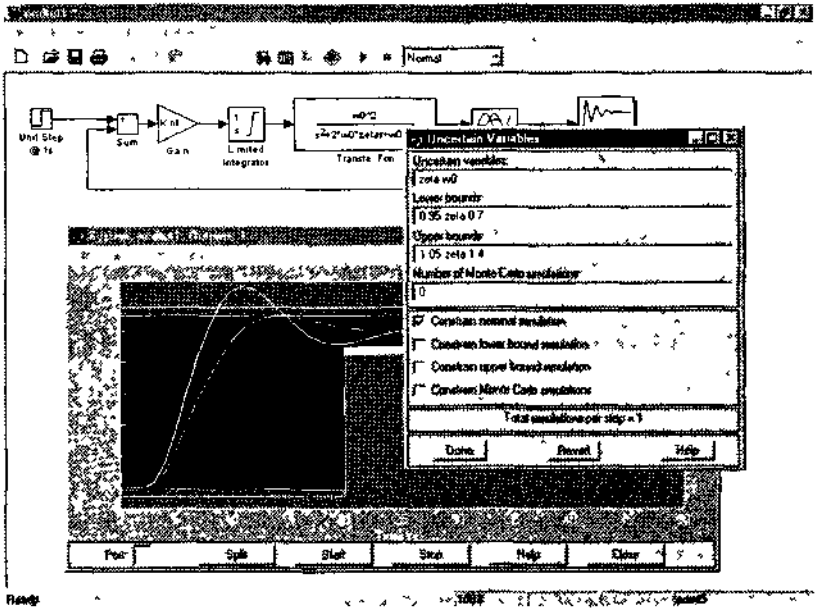


Рис. 12.8. Пример задания неопределенных переменных

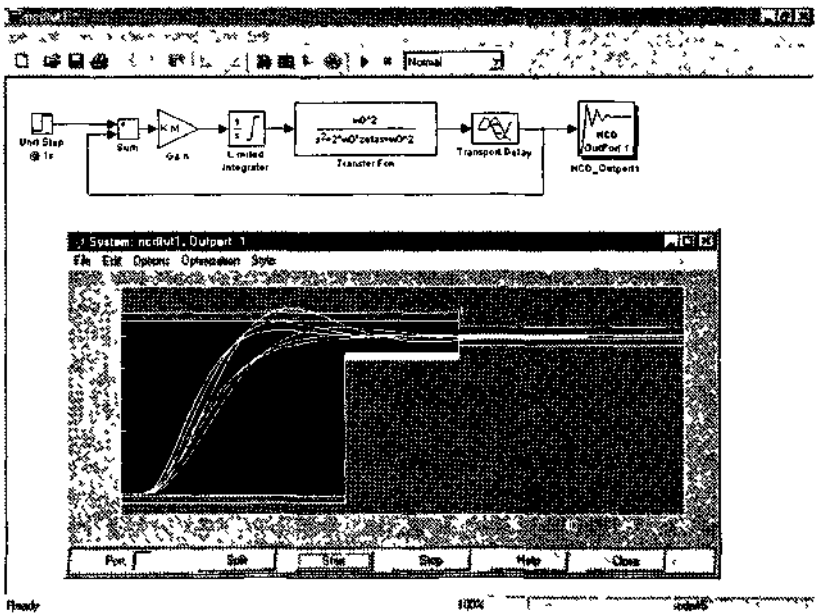


Рис. 12.9. Иллюстрация процесса оптимизации при наличии неопределенных параметров

Нажмем для этого кнопку Start в панели инструментов окна модели Simulink, и результат выполнения оптимизации появится на временной диаграмме (рис. 12.9).

На данном рисунке две светлые кривые соответствуют переходным процессам в системе в том случае, когда неопределенные параметры принимают нижнее и верхнее граничные значения, а коэффициент Kint — начальное значение, две более темные кривые соответствуют переходным процессам для тех же граничных значений, но уже при оптимальном Kint. Теперь это оптимальное значение становится другим:

```
>> Kint
Kint =
    0 1403
```

Отметим, что удалить графики процессов в окне блока NCD Output можно с помощью пункта меню Edit ▸ Delete plots (или одновременным нажатием клавиш Ctrl+X).

Меню окна блока NCD Output

Кратко остановимся на других элементах 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 этих команд, относятся к области заданных временных ограничений, отображаемых в основном окне рассматриваемого блока.

Меню Edit (Редактировать) содержит уже рассмотренные команды Edit constraint и Delete plots, а также команду Undo для отмены предыдущего действия.

Меню Options (Параметры) содержит пункты:

- Initial response (Начальный отклик). Выбор данного пункта приводит к выводу (в основном окне блока NCD Output) отклика исследуемой системы при начальных значениях ее параметров.
- Reference input (Задающий вход). Выбор данного пункта приводит к открытию диалогового окна Reference signal for output (рис. 12.10), в котором можно указать параметры входного сигнала системы для вывода соответствующего графика. Никакого влияния на процессы моделирования и оптимизации данные па-

раметры не оказывают (на рис. 12.10 приведены установки по умолчанию).

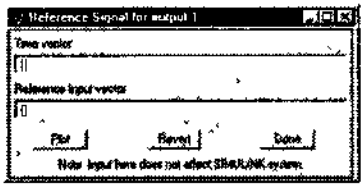


Рис. 12.10. Окно задания параметров задающего воздействия

- Step response (Характеристики переходного процесса). Выбор этого пункта приведет к открытию диалогового окна Step response (рис. 12.11), в котором можно задать параметры переходного процесса, такие как:
  - длительность (Settling time);
  - время нарастания (Rise time);
  - максимальное перерегулирование (Percent overshoot);
  - максимальное «недорегулирование» (Percent undershoot);
  - уровни, на которых определены данные характеристики (Percent settling, Percent rise, в процентах);
  - начальное и конечное время моделирования (Step time и Final time);
  - начальное и желаемое конечное значения выходного сигнала (Initial output и Final output).

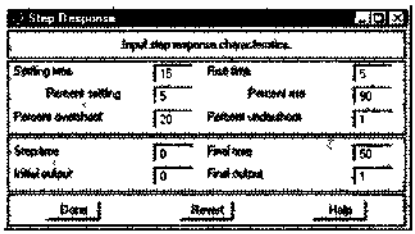


Рис. 12.11. Окно задания характеристик переходного процесса

- Time range (Временной диапазон) Выбор данного пункта приводит к открытию диалогового окна Time Axis Limits (рис 12.12), в котором можно задать или изменить диапазон времени моделирования и метку оси времени, то есть параметры оси абсцисс.

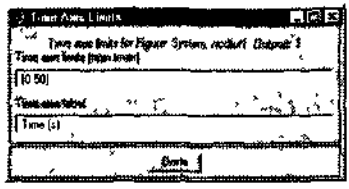


Рис. 12.12. Окно задания временного диапазона графика

- Y-Axis (Ось Y). То же для оси ординат (рис. 12 13).

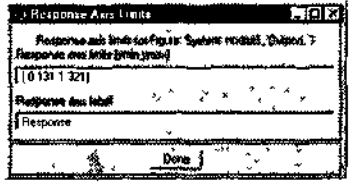


Рис. 12.13. Окно задания диапазона значений оси ординат

- Refresh (Обновить) — перерисовать все временные ограничения.

Меню Optimization (Оптимизация) содержит рассмотренные пункты Parameters и Uncertainty, а также пункт Start (Старт), выбор которого запускает процесс моделирования и оптимизации системы (аналогично нажатию кнопки Start в окне Simulink или в окне блока NCD Output). Пункт Stop (Стоп) останавливает процесс моделирования (аналогично нажатию кнопки Stop в окне блока NCD Output).

Наконец, меню Style (Стиль). Здесь имеются следующие пункты:

- Grid — установка сетки в области заданных ограничений.
- Snap — при установке данного параметра линии временных ограничений можно проводить не под любым углом к оси абсцисс, а только под углом, кратным 22.5 градусам.
- Hot-key help — вывод информации о горячих клавишах и их комбинациях.
- Readme.m — вывод файла справки об окне блока NCD Output

Из пяти кнопок окна NCD Output четыре (Start, Stop, Help, Close) в дополнительных пояснениях не нуждаются. При нажатии кнопки Split (Расщепить) предварительно выбранная ограничивающая линия расщепляется на две равные по длине половинки с возможностью редактирования отдельно каждой из них

Настройка параметров PID-регулятора

Другой пример использования блоков NCD Blockset содержится в файле ncddemo1. Запустив этот файл, в окне Simulink мы увидим модель системы (рис. 12.14).

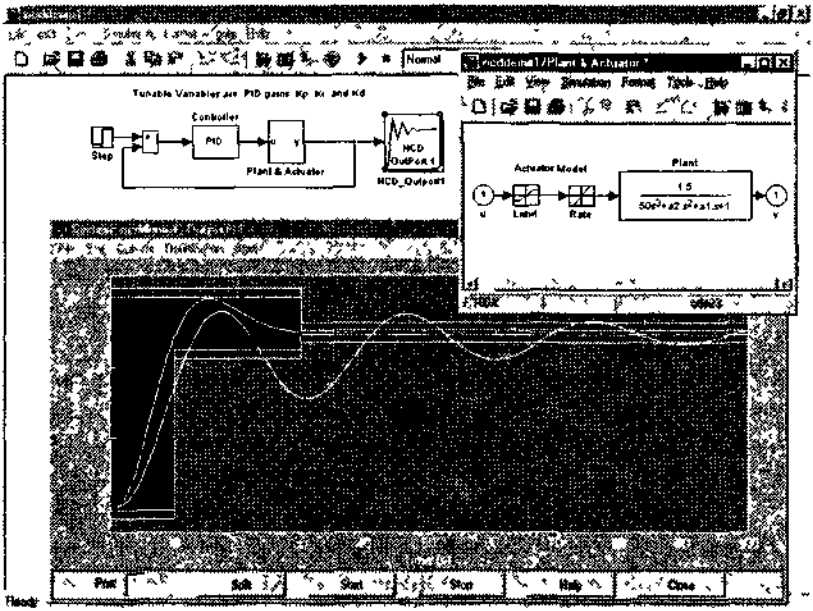


Рис. 12.14. Моделирование PID-регулятора

Основными элементами замкнутой системы PID-регулятора являются:

- объект регулирования (блок Plant&Actuator);
- PID-регулятор (Controller);
- цепь обратной связи и узел сравнения.

Кроме того, в модель входит блок задающего воздействия (в виде единичного скачка) Step и блок NCD Output (NCD OutPort 1). Раскроем блок Plant&Actuator двойным щелчком мыши. Окно параметров этого блока показано на рис. 12.14.

В состав объекта регулирования входит нелинейность с уровнями ограничения –2, 2 (блок Limit), блок динамического ограничения коэффициента усиления (Rate), осуществляющий 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

$$W(p)=\frac{1\,5}{50p^3+a_2\cdot p^2+a_1\;\;p+1},$$

где коэффициент a_2 может принимать значения в диапазоне [40, 50] с номинальным значением $a_2 = 43$, а коэффициент a_1 — в диапазоне от [0.5, 3] с номинальным значением $a_1 = 1.5$.

Постановка задачи оптимизации в данном случае такова: при заданной структуре объекта управления и известных неопределенностях его параметров найти значения коэффициентов Kp, Ki и Kd регулятора, при которых в представленной замкнутой структуре переходный процесс будет иметь параметры, заданные по умолчанию.

Отметим, что моделирование в данном случае будет проводиться для номинальных значений коэффициентов a_1 и a_2 и при начальных значениях параметров оптимизации Kp = 0.63, Ki = 0.0504, Kd = 1.9688. Такие значения выбраны в соответствии с методикой настройки PID-регуляторов Зиглера–Николса (Ziegler-Nichols method), согласно которой:

- коэффициенты Ki и Kd устанавливаются равными нулю, а коэффициент Kp увеличивается до тех пор, пока система не потеряет устойчивость;
- предельное значение Kp обозначается как Ku, а период автоколебаний — как Pu;
- задаются следующие значения коэффициентов регулятора:

```
Kp = 3  Ku/5,     Ki = 6 · Ku/(5  Pu)     Kd = 3 · Ku  Pu/40
```

Запустим процесс оптимизации, нажав кнопку Start. Полученный результат отражен на рис. 12.14.

Найденные оптимальные значения (в командном режиме работы MATLAB) при этом равны.

```
>> Kp
Kp =
    1 3365
>> Ki
Ki =
    0 1548
>> Kd
```

```
Kd =
    8 3317
```

и существенно отличаются от начальных.

Проведем теперь оптимизацию, учитывая интервальную неопределенность коэффициентов a_1 и a_2, 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 показан на рис. 12.15 (светлые линии соответствуют исходной системе при граничных значениях неопределенных параметров, темные — оптимизированной).

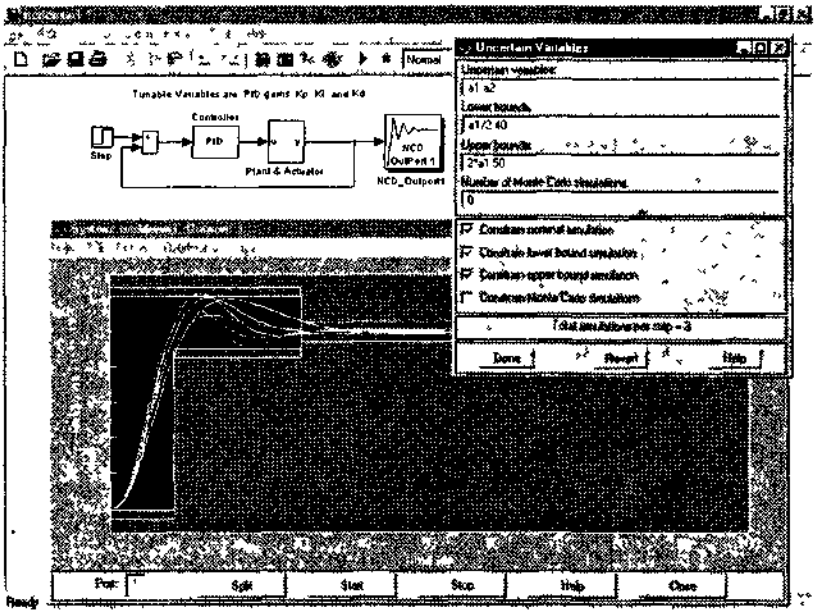


Рис. 12.15. Иллюстрация процесса оптимизации при неопределенных параметрах a1 и a2

В командном режиме вычислений находим:

```
>> Kp
Kp =
    1 6740
>> Ki
Ki =
    0 1273
>> Kd
```

```
Kd =
    8.2597
```

По сравнению с предыдущим случаем здесь имеются определенные отличия.

Процесс исследования системы можно продолжить, вводя, например, дополнительные ограничения или изменяя введенные и т. п.

Настройка параметров комплексного регулятора

Следующим примером (отраженным в файле ncddemo2) является пример настройки параметров регулятора сложной структуры. При открытии данного файла в окне Simulink появится модель системы (рис. 12.16).

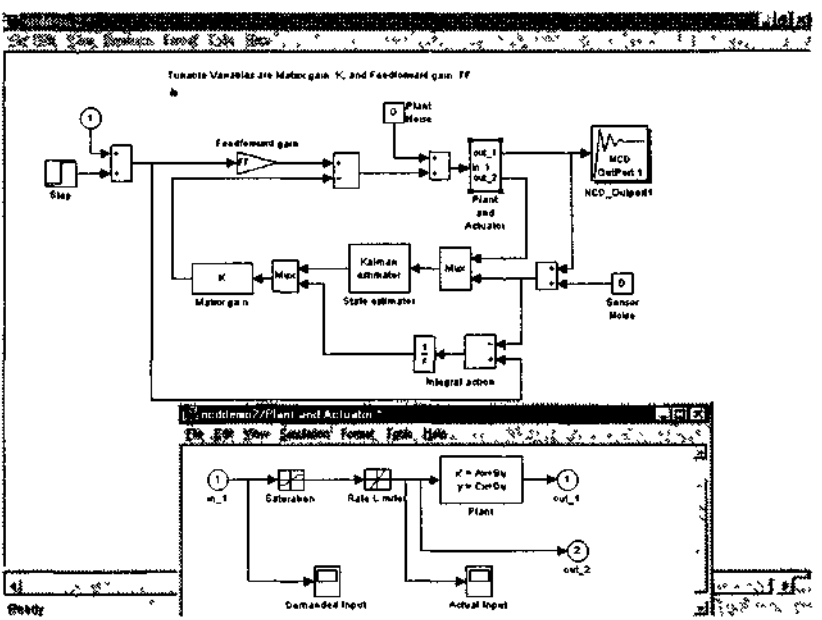


Рис. 12.16. Модель системы регулирования

Объект регулирования (Plant and Actuator) представляет собой (см. рис. 12.16 внизу) последовательно соединенные блоки: нелинейность с функцией ограничения (Saturation), нелинейность динамического ограничения коэффициента усиления — блок Rate Limiter (см. выше)

и линейное динамическое звено (Plant), описание которого задано через переменные состояния.

Характеристики звеньев: уровни ограничения −5 и +5, динамическое ограничение определяется диапазоном [−10, 10], описание линейного динамического звена имеет вид:

$$\dot{x} = \begin{bmatrix} -1\ 0285 & 0\ 9853 & -0\ 9413 & 0\ 0927 \\ -1\ 2903 & -1\ 0957 & 2\ 8689 & 4\ 7950 \\ 0\ 1871 & -3\ 8184 & -2\ 0788 & -0\ 9781 \\ 0\ 4069 & -4\ 1636 & 2\ 5407 & -1\ 4236 \end{bmatrix} x + \begin{bmatrix} 0 \\ 6\ 6389 \\ 0 \\ 0 \end{bmatrix} u = Ax + Bu,$$

$$y = \begin{bmatrix} -1\ 7786 & 1\ 1390 & 0 & -1\ 0294 \end{bmatrix} x = Cx$$

Предполагается, что все элементы матрицы A здесь могут изменяться от половины до двух своих номинальных значений.

Регулятор в данном случае имеет сложную структуру: в его состав входит как обычный И-регулятор (блоки Integral action), так и преобразователь 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, при которых:

- перерегулирование не превышает 20%;
- время установления — не более 1 с;
- длительность переходного процесса — не более 3 с.

Вы можете просмотреть установки исходных параметров так, как это описывалось ранее. Результат моделирования представлен на рис. 12.17.

При исходных 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 можно получить в режиме командной строки MATLAB:

```
K =
   -1 0585     0 9378     0 0188     0 2413     0 0786
>> FF
FF =
    1 2729
```

Заметим, что при установленных параметрах (рис 12 17) найденное решение соответствует номинальным значениям элементов матрицы *A* Для исследования поведения системы с учетом неопределенности данной матрицы необходимо выполнить действия, описанные в предыдущем примере

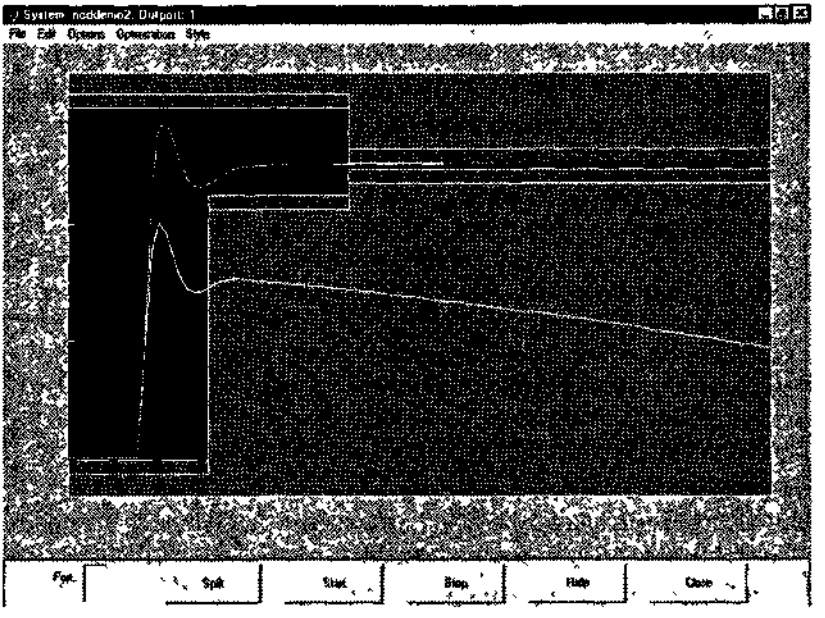


Рис. 12.17. Иллюстрация процесса оптимизации

Настройка параметров PI-регулятора для многомерного объекта

Наконец, еще одним примером (отраженным в файле nddemo3) является пример настройки параметров пропорционально-интегрального регулятора (PI-регулятора) для многомерного объекта. В качестве такого объекта рассмотрен газотурбинный двигатель. Его математическое описание представлено в стандартной форме через переменные 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. 12.18.

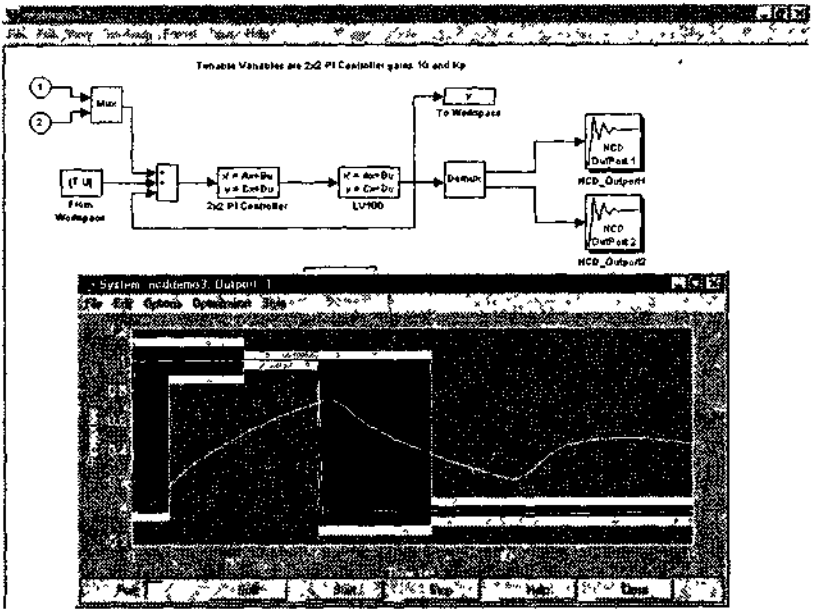


Рис. 12.18. *Система с многомерным объектом регулирования*

В данном случае параметрами оптимизации являются матрицы коэффициентов Ki и Kp двухканального PI-регулятора (блок 2 × 2 PI Controller). Условия настройки и ограничения могут быть проконтролированы описанными выше средствами и видны из положений граничных линий. Задающие воздействия имеют вид прямоугольных импульсов.

Инициализация системы (двойным щелчком мыши на блоке ncd3init), а затем запуск процесса моделирования и оптимизации приводят к требуемому результату (в данном примере поиск оптимальных значений параметров проходит существенно дольше, чем в предыдущих примерах, из-за большей размерности и сложности задачи). Полученные результаты, также отображенные на рис. 12.18, следует признать очень качественными (здесь оптимизация проведена для номинальных значений элементов матрицы A).

Найденные оптимальные значения таковы:

```
>> K1
K1 =
   12 5209   -2 5168
   46 2438   51 7909
>> Kp
Kp =
    1 6317    6 0054
  127 0848   58 4045
```

Отметим, что в файлах ncddemo4, ncdtut2, ncdtut2old содержатся еще три примера, иллюстрирующие работу и применение блоков NCD Blockset. Дополнительную информацию можно получить, используя команду help NCD.

Особенности решаемых задач

При решении с помощью пакета NCD Blockset различных задач оптимизации следует иметь в виду следующие особенности этого пакета:

- Пакет может использоваться для решения задач оптимизации переходного процесса, идентификации, настройки параметров системы с использованием квадратичного критерия качества для случаев, когда все сигналы в исследуемой системе (на этапе ее оптимизации) являются детерминированными и 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 — 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» в рассматриваемом пакете решается однократным заданием повторяющегося параметра среди других оптимизируемых параметров.
- 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 параметров оптимизации) требуемые нули и полюса параметрами оптимизации с заданием, разумеется, их предельных 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 имеем дело с повторяющимся параметром, во втором — нет, но в первом случае задание ограничений представляется более простым.

В целом надо отметить, что, несмотря на очень малое число блоков этого пакета расширения, он решает исключительно важную и сложную задачу оптимизации нелинейных систем.

Глава 13. Пакет расширения Fixed-Point Blockset

- Библиотека пакета Fixed-Point 3.0
- Основные операции пакета Fixed-Point
- Подсистемы и графический интерфейс Fixed-Point

Библиотека пакета Fixed-Point

Доступ к библиотеке Fixed-Point

Мало известный российским пользователям пакет Fixed-Point Blockset 3.0 является расширением пакета Simulink и предлагает разработчикам дискретных систем эффективные средства вычислений с фиксированной точкой. Этот формат вычислений, называемый также F-форматом, поддерживается на аппаратном уровне компьютера, а потому моделирование в нем выполняется предельно быстро.

В этом пакете имеется ряд мощных средств для выполнения математических и логических операций над данными в формате с фиксированной точкой (F-формате), набор блоков для преобразования обычного формата сигнала (V-формата — с плавающей точкой двойной точности) в F-формат и наоборот, а также простые и удобные средства преобразования аналоговых сигналов в цифровые (дискретные) и наоборот.

Доступ к библиотеке пакета выполняется обычным способом — через браузер библиотек Simulink (рис. 13.1).

Основной раздел библиотеки

Как видно из рис. 13.1, библиотека пакета Fixed-Point состоит из одного раздела с блоками пакета и встроенного в него другого раздела. Рисунок 13.2 показывает окно с полным набором блоков основного раздела

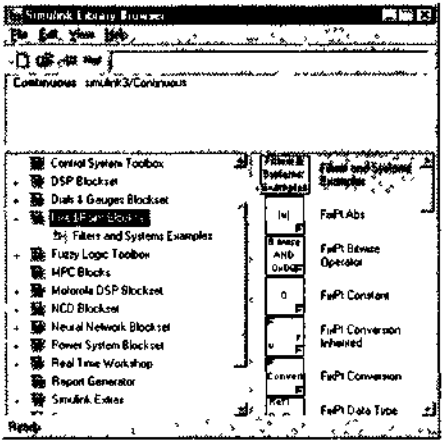


Рис. 13.1. Доступ к библиотеке Fixed-Point Blockset

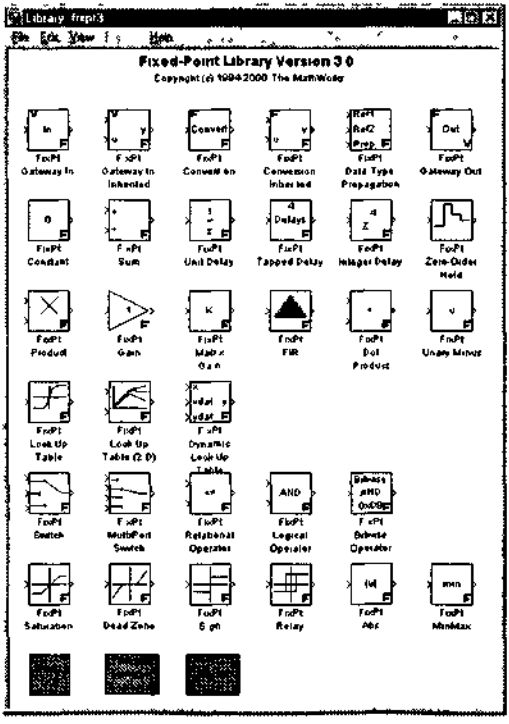


Рис. 13.2. Окно основного раздела библиотеки Fixed-Point Blockset

В этот раздел входит 32 блока, их которых подавляющее большинство выполняет практически те же функции, что и блоки ранее рассмотренной основной библиотеки Simulink. Однако между ними имеется существенное различие — блоки пакета Fixed-Point работают в дискретном режиме и используют операции с фиксированной точкой. Поэтому, чтобы не путать их с подобными блоками основной библиотеки, у этих блоков в пиктограммах имеется большая буква F в правом нижнем углу.

Помимо указанных блоков в окне рис. 13.2 внизу видны еще 3 значка:

- FixPt GUI — доступ к специальному интерфейсу пользователя пакета;
- Filters&System Examples — демонстрационные примеры на построение дискретных фильтров и систем;
- Demos — демонстрационные примеры.

Раздел демонстрационных примеров Filters&System Examples

Раздел Filters&System Examples библиотеки пакета Fixed-Point имеет свое окно (рис. 13.2), в котором расположены значки запуска демонстрационных примеров, а под ними — значки блоков, которые используются в этих примерах.

Надо отдать должное разработчикам пакета — он содержит множество поучительных и полезных примеров, явно ориентированных на самостоятельное знакомство с обширными возможностями этого пакета. В то же время нельзя не отметить, что это, конечно, пакет специального назначения и подавляющее большинство задач моделирования можно решать, не обращаясь к нему.

Получение информации о блоках

У большинства блоков пакета Fixed-Point в основном разделе библиотеки (рис. 13.2) вид пиктограммы и название явно указывают на их назначение. Например, блок с надписью в пиктограмме |u| и названием FixPt Abs, конечно же, является блоком вычисления абсолютного значения входного сигнала U.

Ряд блоков служат для выполнения стандартных математических операций. Это блоки задания константы FixPt Constant, масштабирования FixPt Gain, умножения FixPt Product, скалярного умножения FixPt Dot Product, сложения Sum FixPt, вычисления знака FixPt Sign и др.

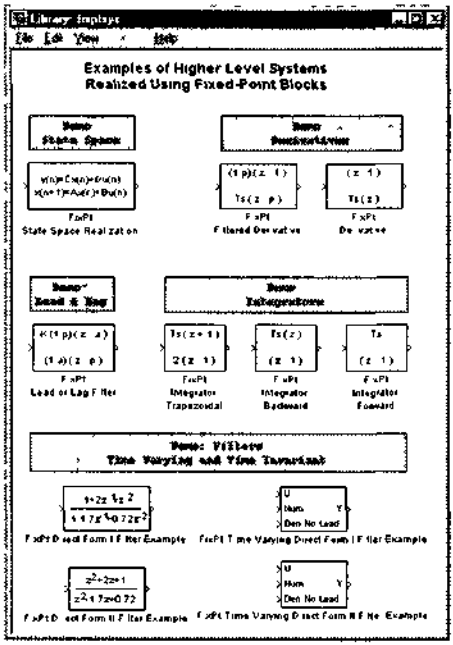


Рис. 13.3. Окно раздела Filters&System Examples библиотеки пакета Fixed-Point

Блоки, названия которых содержат слово Conversion, служат для преобразования данных, а блоки со словом Delay — для задержки сигналов.

Многие из блоков выполняют функции нелинейного преобразования сигналов и характеризуются нелинейной статической передаточной характеристикой $Y=f(U)$. Такие блоки распознаются по характерному упрощенному изображению этой характеристики. К ним относятся блок ограничения FixPt Saturation, блок с зоной нечувствительности FixPt Dead Zone и др.

Блоки, представленные вторым разделом библиотеки Fixed-Point (рис. 13.3), характеризуются не только вполне очевидным из их имен функциональным назначением, но и операторной передаточной характеристикой (функцией от s). Мы не будем многократно приводить эту функцию, поскольку она явно указана в пиктограмме блока и детально описана в окнах параметров этих блоков. Множество демонстрационных примеров позволяют усвоить тонкости применения этих блоков

Кстати говоря, блоки этого раздела не являются полностью самостоятельными блоками. Это маскированные подсистемы, и их графические модели открыты для просмотра и модификации.

По назначению, функциям и системе параметров эти блоки в целом соответствуют уже описанным блокам основной библиотеки. Различие только в том, что блоки пакета Fixed-Point всегда дают дискретный выходной сигнал, равномерно квантованный по времени. Учитывая это, а также схожесть интерфейса большинства блоков этого пакета с интерфейсом блоков основной библиотеки, мы не будем повторять описание уже известных параметров блоков пакета Fixed-Point.

Основные операции пакета Fixed Point

Задание и умножение константы

На рис. 13.4 дан простейший пример применения блоков пакета Fixed-Point — задание константы 5.2 и ее умножение на 2.0. Для контроля выходов блоков используется дисплей из раздела Sinks основной библиотеки Simulink. После запуска модели командой Start можно наблюдать за работой модели по показаниям цифрового дисплея.

На рис. 13.4 показаны также окна с параметрами блоков задания константы и умножения на константу (масштабирования). Нетрудно заметить, что их интерфейс ничем не отличается от описанного ранее интерфейса подобных блоков основной библиотеки. Однако, как правило, параметров у блоков пакета Fixed-Point несколько больше, в основном за счет необходимости указания форматов чисел для входных и выходных сигналов, а также за счет указания погрешностей при округлении. Это плата за более быстрое выполнение операций блоками данного пакета. Ввиду очевидности большинства параметров блока мы не будем далее заострять на этом внимание.

Нелинейные преобразования

Рисунок 13.5 показывает преобразование синусоидального сигнала (источник Sine Wave из раздела Sources основной библиотеки Simulink) рядом нелинейных блоков пакета Fixed-Point. Для контроля выходных сигналов используется виртуальный осциллограф Scope из раздела Sinks основной библиотеки Simulink.

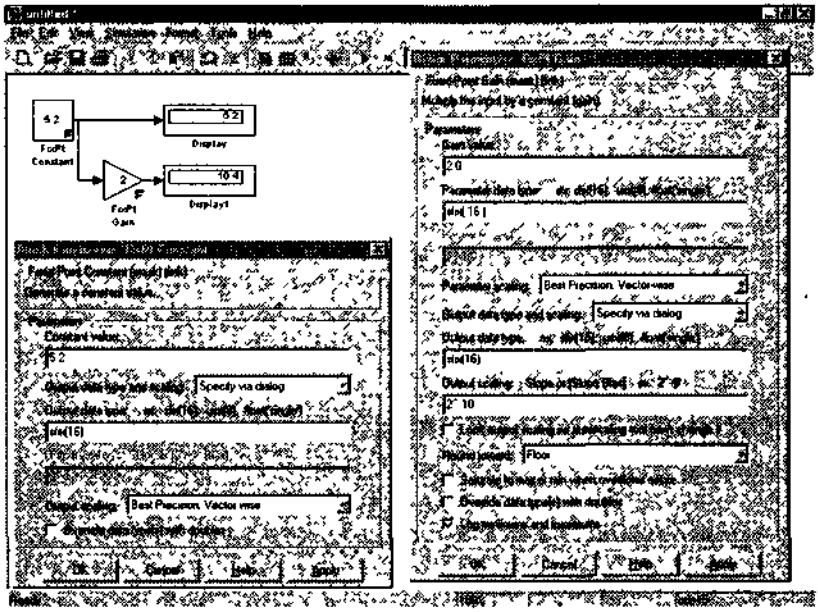


Рис. 13.4. Пример задания константы и ее умножения

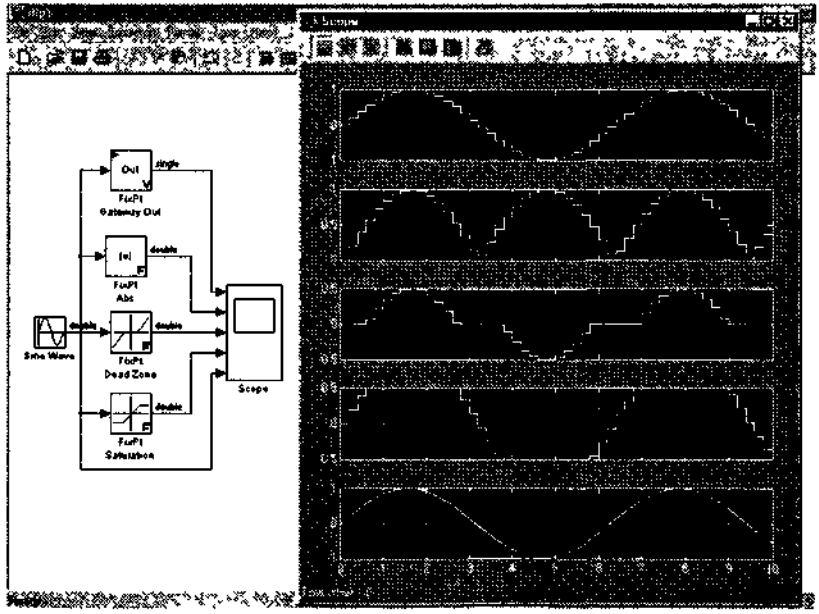


Рис. 13.5. Задание синусоиды и ее нелинейное преобразование

Обратите внимание на то, что входной сигнал задан как аналоговый, но на выходах этих блоков он уже оказывается дискретным. Таким образом, блоки пакета Fixed-Point являются естественными квантующими устройствами, преобразующими аналоговые сигналы в цифровые. Квантование происходит равномерно во времени, шаг его задается эталонным временем Sample time (в этом примере его значение равно 0.2 и задается в параметрах блоков). Чем меньше это время, тем менее заметны вызванные квантованием ступеньки. Читателю рекомендуется просмотреть окна параметров всех блоков этого примера.

Простые математические операции

Примеры выполнения ряда математических операций приведены на рис. 13.6. Здесь задана квантованная синусоида (Sample time = 0.2 в блоке Sine Wave) и проверяется реакция на нее блоков унарного минуса (дает синусоиду в противофазе), фильтрации типа FIR, умножения на константу 2 (дает удвоение амплитуды синусоиды) и вычисления знака (превращает синусоиду в меандр). Для контроля сигналов используется осциллограф.

Нетрудно заметить, что и в этом примере поведение блоков вполне предсказуемо и соответствует представлениям, полученным из описания соответствующих блоков основной библиотеки.

Задержка сигналов

В состав основного раздела библиотеки пакета Fixed-Point входит несколько блоков временной задержки. Рисунок 13.7 поясняет их применение для задержки квантованного сигнала.

По приведенным на рис. 13.7 осциллограммам легко судить о функциях этих блоков.

Суммирование и умножение двух синусоидальных сигналов

Начиная с этого примера (рис. 13.8) мы приведем несколько сравнительно простых демонстрационных примеров, входящих в пакет Fixed-Point. Имена файлов не указываются, поскольку они видны в заголовке окна модели каждого примера.

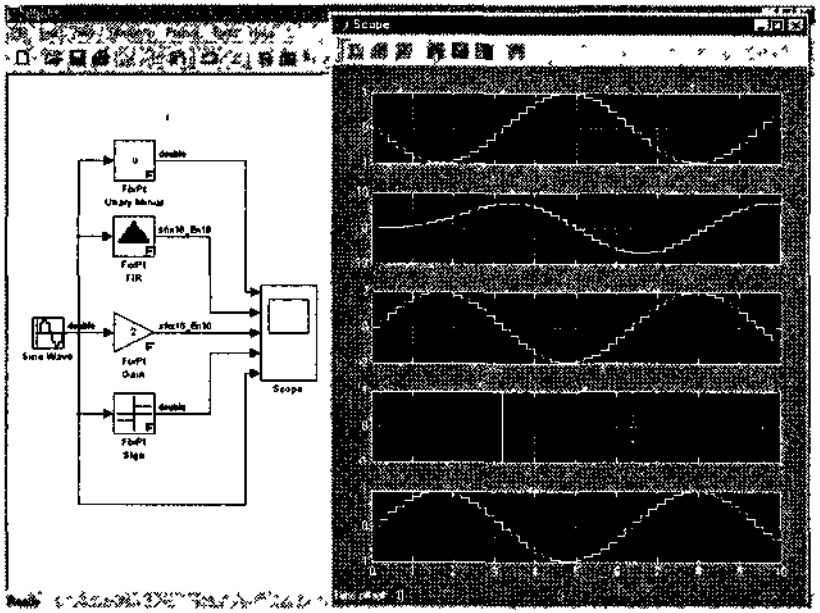


Рис. 13.6. Воздействие блоков простых математических операций на синусоиду

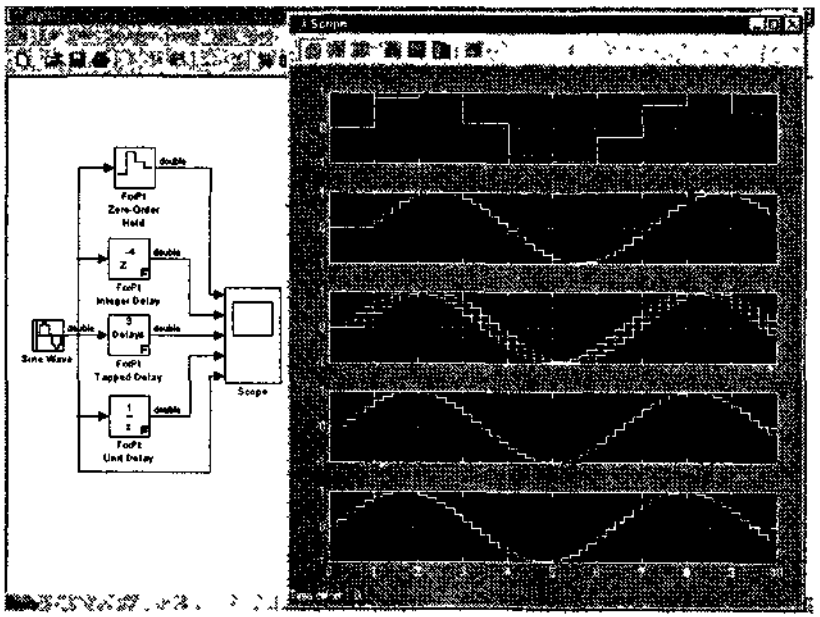


Рис. 13.7. Примеры задержки сигнала средствами пакета Fixed-Point

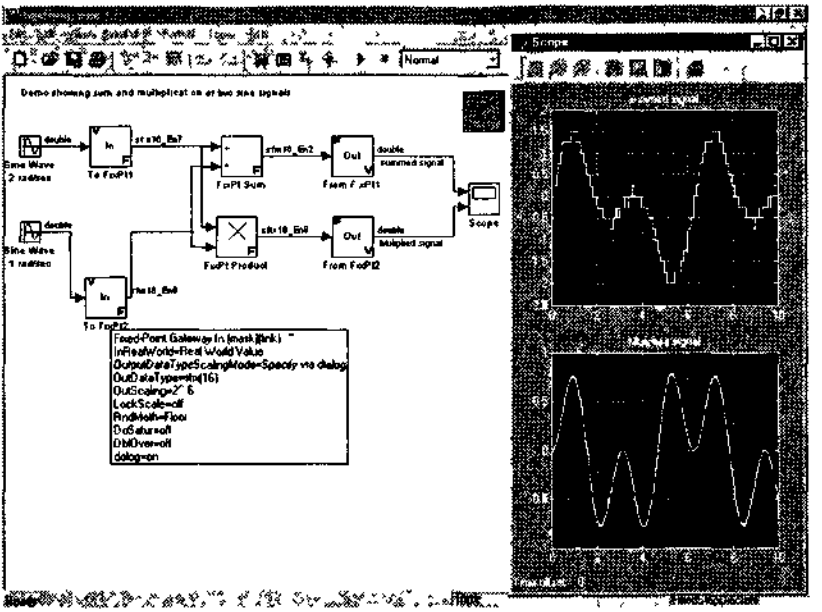


Рис. 13.8. Суммирование и умножение двух синусоидальных сигналов

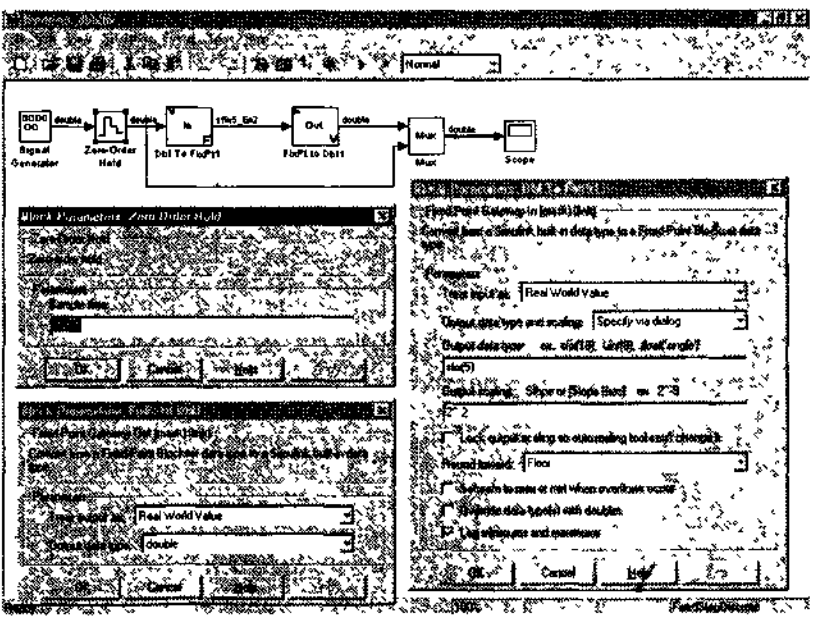


Рис. 13.9. Пример задержки нулевого порядка с преобразованием форматов выходного сигнала

В структурной диаграмме модели хорошо видна роль блоков ввода In и вывода Out пакета Fixed-Point, а также блоков сложения и умножения двух синусоидальных сигналов.

Задержка нулевого порядка

Рисунок 13.9 иллюстрирует работу блока задержки нулевого порядка Zero Order Hold (из основной библиотеки) и двух блоков преобразования из пакета Fixed-Point.

Ввиду редкого применения этих блоков мы не будем обсуждать их подробно, тем более что большинство пользователей удовлетворится представленными на рис. 13.9 установками по умолчанию. Осциллограммы, иллюстрирующие работу этого примера, даны на рис. 13.10.

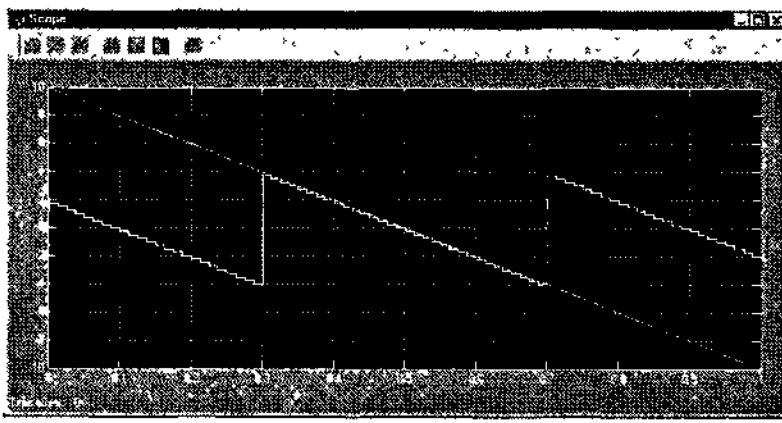


Рис. 13.10. Осциллограммы модели рис. 13.9

Преобразования вида V-F, F-F и F-V

Среди блоков пакета Fixed-Point есть несколько блоков, осуществляющих преобразования данных из обычного V-формата в формат с фиксированной точкой F. На рис. 13.11 показана модель, хорошо иллюстрирующая применение преобразователей вида V-F, F-F и F-V. Обратите внимание на то, что у таких блоков формат входа указан буквой в верхнем левом углу пиктограммы блока, а формат выхода — в правом нижнем углу. Осциллограммы этой модели также приведены на рис. 13.11.

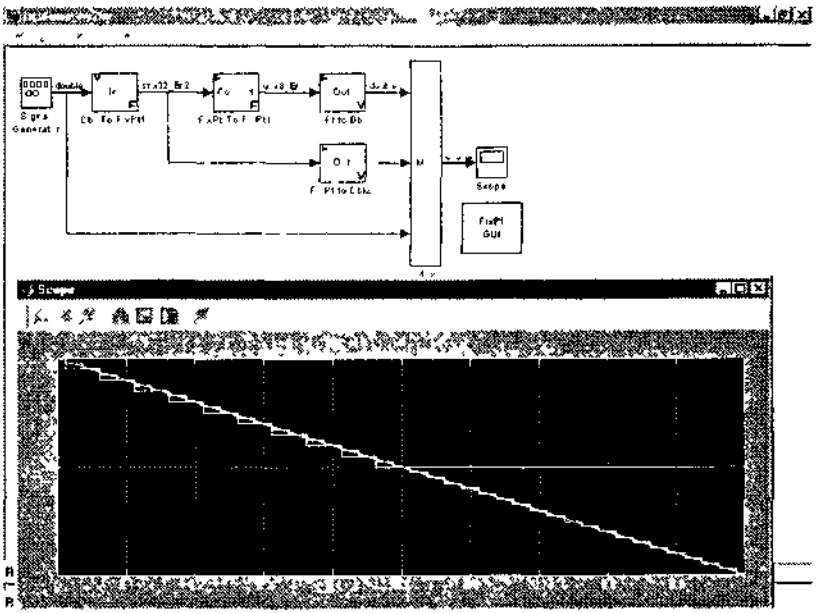


Рис. 13.11. Преобразования вида V-F, F-F и F-V

Подсистемы и графический интерфейс пакета Fixed-Point

Цифровой программный контроллер

Рисунок 13.12 демонстрирует модель цифрового программного контроллера на основе подсистемы Fixed-Point-процессора.

В этой модели сигнал в виде прямоугольных симметричных импульсов (меандра) проходит длинную цепь преобразований: он задерживается, преобразуется в цифровой сигнал, проходит подсистему программного контроллера, преобразуется вновь в аналоговый сигнал, корректируется и подается в качестве сигнала отрицательной обратной связи на вход (для этого на входе имеется двухвходовой сумматор).

Осциллограммы блока, представленные на рис. 13.12, показывают заметные искажения формы сигнала по сравнению с идеальным входным сигналом тестового генератора. Однако с практической (да и технической) позиции можно говорить о вполне удовлетворительной форме восстановления сигнала, которая получается, несмотря на многочисленные и сложные его преобразования.

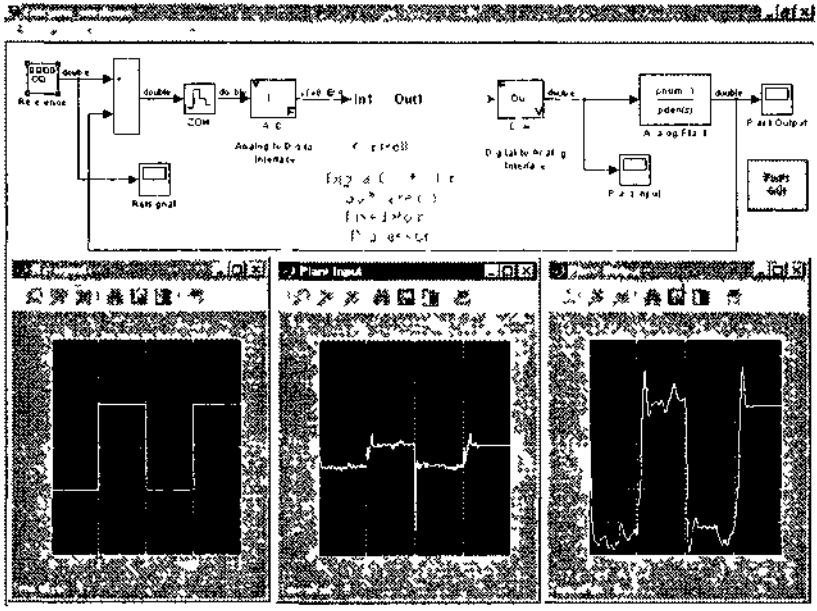


Рис. 13.12. Модель цифрового программного контроллера

Установка параметров моделирования

На этом примере мы обсудим важный вопрос установки параметров моделирования. Модели с блоками расширения Fixed-Point часто моделируются с постоянным шагом во времени. Пример установки фиксированного шага моделирования дан на рис. 13.13.

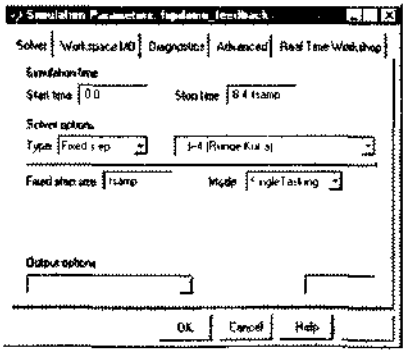


Рис. 13.13. Установка параметров моделирования

Здесь задан фиксированный шаг (условный) моделирования tsamp. Обратите внимание на то, что конечное время моделирования задано через шаг tsamp. Это позволяет выхватить пару периодов сигналов из их потока. Для моделирования использован хорошо известный метод Рунге-Кутта 4-го порядка

Подсистемы модели

Двойным щелчком мыши на подсистеме цифрового программного контроллера можно вывести окно с диаграммой этой подсистемы (рис 13.14).

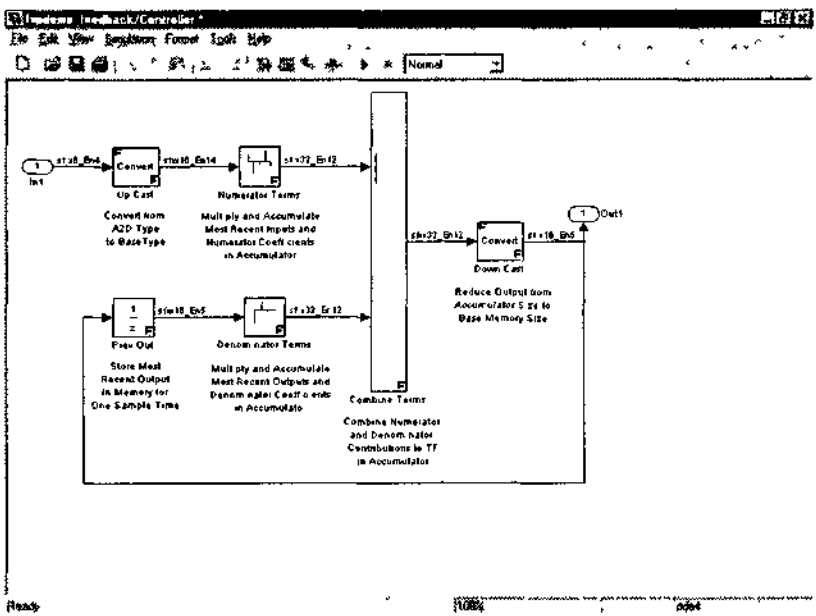


Рис. 13.14. Подсистема цифрового программного контроллера

Читатель, уже имеющий опыт в разработке цифровых устройств, может заинтересоваться деталями работы этой подсистемы.

Интерфейсный блок пакета Fixed-Point

Пакет Fixed-Point имеет специальный интерфейсный блок, который можно поместить в любую модель. Этот блок автономный — он никуда не подключается, но тем не менее к нему подводятся все сигна-

лы, действующие в модели. Блок предназначен для углубленного контроля за типами и значениями сигналов и построения их детальных графиков, иногда более приемлемых, чем осциллограммы, получаемые от виртуальных осциллографов.

На рис. 13.12 имеется интерфейсный блок с именем FixPt GUI (Fixed-Point Graphic User Interface). Двойным щелчком мыши на этом блоке можно открыть окно интерфейса (рис. 13 15).

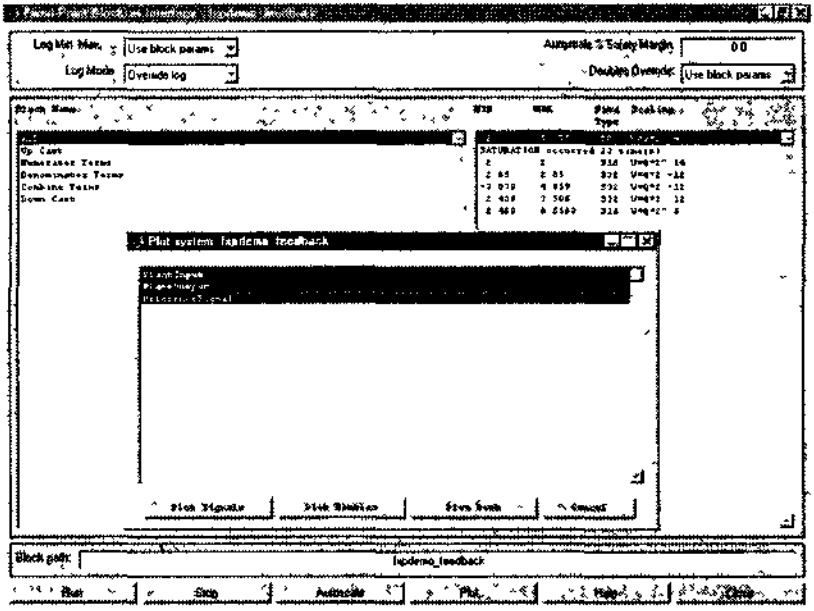


Рис. 13.15. Интерфейсный блок FixPt GUI

Этот интерфейс явно предназначен для работы на дисплеях с большим экраном и высоким разрешением, которые используются в системах автоматического проектирования (САПР).

На экранах дисплеев с размерами экрана по диагонали в 15–17 дюймов и типовом разрешении 800 × 600 точек (пикселов) надписи на блоке хотя и вполне различимы, но получаются явно мелковатыми.

Зато интерфейс позволяет одновременно оценивать параметры десятков блоков. В нашем примере список блоков содержит шесть наименований. Список блоков текущей модели (с ее подсистема-

ми) представлен в левой части окна, а параметры выделенного блока отображаются в правой части окна. Двойной щелчок мышью на элементе списка выводит окно параметров соответствующего блока.

В целом интерфейс отличается наглядностью и не требует подробного описания. Внизу окна расположен ряд кнопок вполне очевидного назначения. С их помощью можно запускать и останавливать моделирование, наблюдая за изменением параметров или за графиками сигналов модели

Для построения графиков сигналов служит кнопка Plot. Она выводит окно для установки режимов представления графиков (одного или нескольких) (рис. 13.15). На рис 13.16 дан пример построения графиков входного и выходного сигналов.

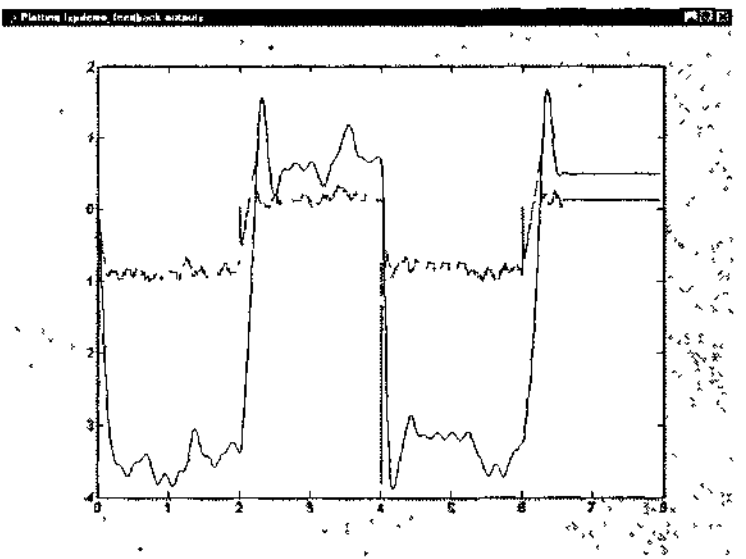


Рис. 13.16. Пример построения графиков средствами графического интерфейса пакета Fixed-Point

Будучи открытыми на весь экран, как это и показано на рис. 13.16, такие графики дают более четкое представление о форме сигналов (в нашем случае на входе Plant Input и на выходе Plant Output), чем осциллограммы

Примеры применения раздела Filters&System Examples

Фильтрация производной

В заключение описания пакета Fixed-Point рассмотрим некоторые примеры из раздела Filters&System Examples. Эти примеры, как правило, довольно громоздки, и мы не ставим целью детальное знакомство с ними. Приведенное ниже описание — это, скорее, обзор проблематики, к которой относятся приведенные примеры, стимулирующий читателя более внимательно ознакомиться с ними.

В примере fxpdemo_diff описана модель, которая обеспечивает фильтрацию производной сигнала. Сам рисунок модели мы не приводим ввиду его громоздкости На рис. 13.17 показаны осциллограммы сигналов этой модели.

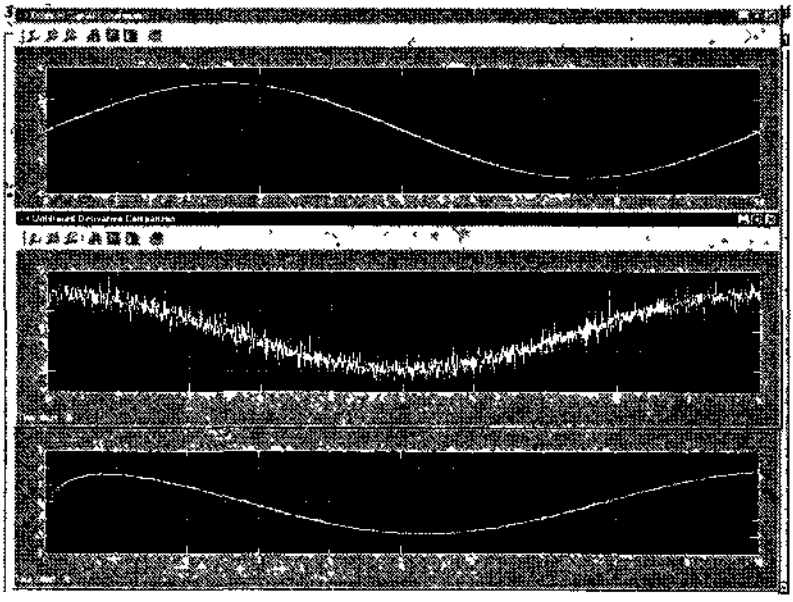


Рис. 13.17. Осциллограммы модели фильтрации производной

Все осциллограммы представлены в сравнительных вариантах. Верхние осциллограммы показывают исходный синусоидальный сигнал и результат его квантования (сигнал на входе модели). За счет боль-

шого числа шагов квантования осциллограммы почти сливаются. Средние осциллограммы показывают производную сигнала с наложенными на нее шумами и результат ее квантования. Нижние осциллограммы (сигнал на выходе модели) иллюстрируют обработку зашумленного сигнала производной и выделение ее из шума (также в варианте квантованного и аналогового сигнала). Высокая эффективность данной модели очевидна.

Рисунок 13.18 показывает окно интерфейса пакета Fixed-Point с данными этого примера. Здесь также показана настройка графического окна на отображение графиков всех сигналов на одном экране.

Графики всех сигналов описываемой модели показаны на рис. 13.19. Можно заметить, что и здесь они отличаются большей детальностью, чем осциллограммы, представленные на рис. 13.17, несмотря на переход к меньшему разрешению дисплея.

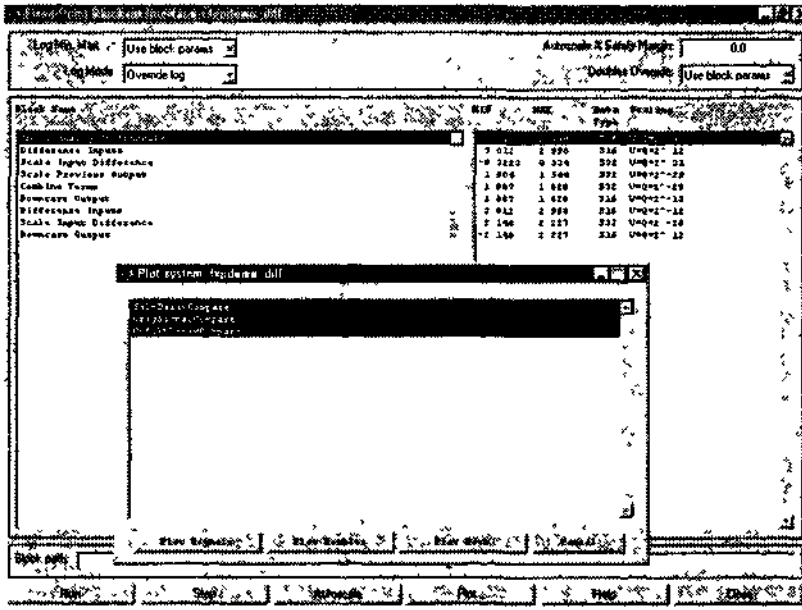


Рис. 13.18. Работа с интерфейсом пакета Fixed-Point для примера на фильтрацию производной

Цифровое интегрирование

Другой пример — fxpdemo_integrate, тоже довольно громоздкий, показывает реализацию модели цифрового интегратора и открытое

окно одного из блоков, выполняющих функции интегрирования в F-формате.

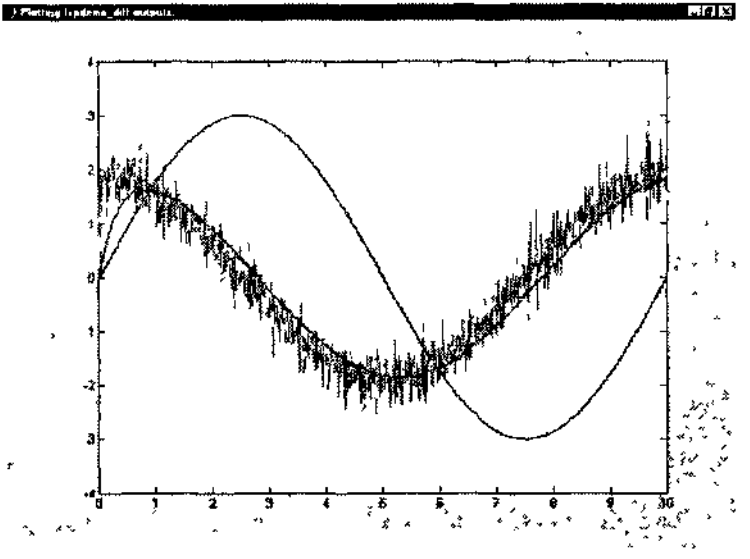


Рис. 13.19. Графики сигналов модели фильтрации производной в одном окне

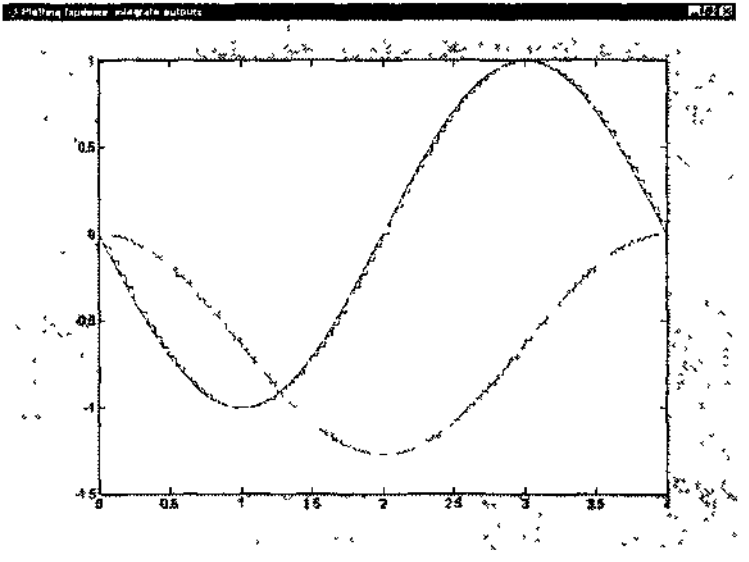


Рис. 13.20. Графики сигналов модели интегратора

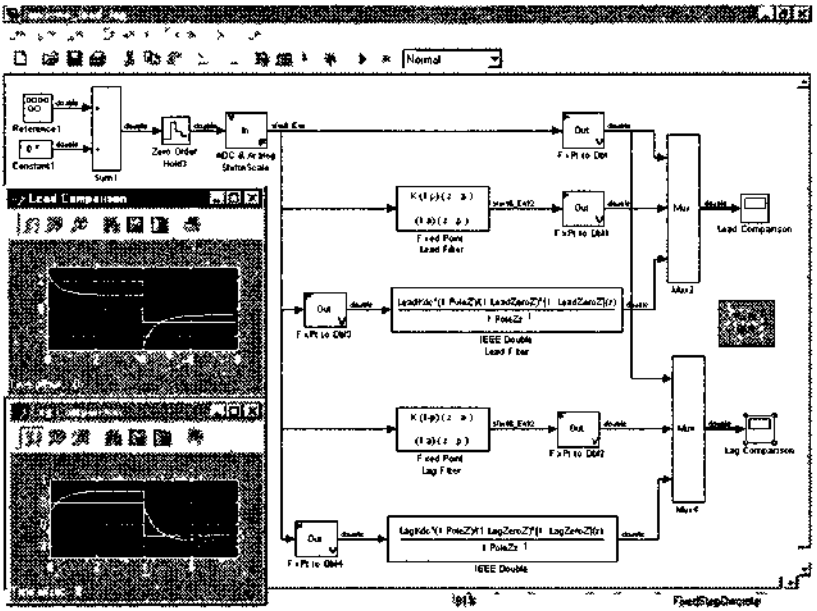


Рис. 13.21. Пример применения блоков Lead и Lag фильтрации в F-формате

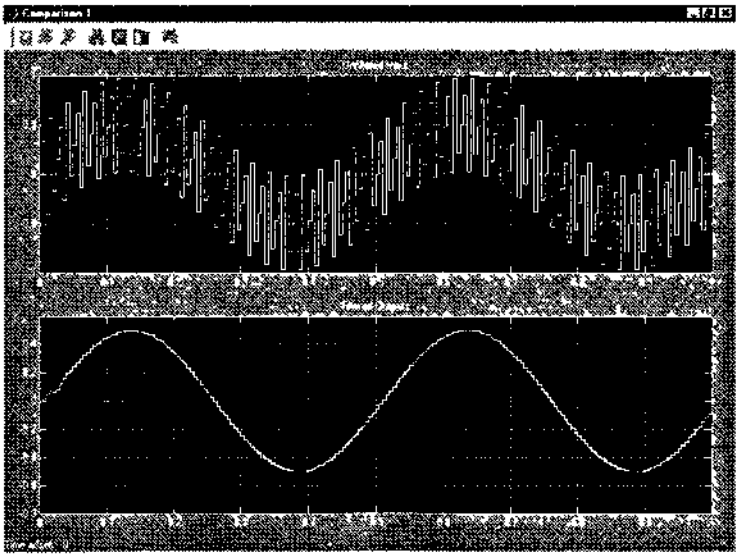


Рис. 13.22. Осциллограммы для трех моделей фильтра Баттерворта

Временные диаграммы работы этой модели, полученные с помощью интерфейса пользователя пакета Fixed-Point, представлены на рис 13 20

Lead- и Lag-фильтрация

Рассмотрим осуществление так называемой Lead- и Lag-фильтрации в F-формате (Lead — опережающий по фазе фильтр, а Lag — запаздывающий). Соответствующие блоки фильтров использованы в модели рис. 13.21 Операторные характеристики этих фильтров представлены выражениями, записанными в пиктограммах блоков (что делает эти фильтры полностью определенными).

Сравнение фильтров Баттерворта

Пример fxpdemo_filterd содержит средства для моделирования фильтров Баттерворта разного типа.

Модель задает три реализации фильтров Баттерворта на основе F-реализации (аналоговый фильтр) и две реализации цифровых фильтров Баттерворта

Рисунок 13 22 показывает сравнительные осциллограммы входного сигнала (сверху) и трех отмеченных реализаций фильтра Баттерворта (снизу) Осциллограммы цифровых фильтров практически совпадают, и различить их невозможно

На рис 13 23 даны осциллограммы, иллюстрирующие фильтрацию сигнала инвариантного по времени цифрового фильтра Баттерфорта Процесс фильтрации начинается с задержкой, и хорошо видны переходные процессы, связанные с этим. Они занимают меньше четверти периода выходного сигнала, практически синусоидального.

Этот пример дает хорошее представление о сложности решаемых с помощью пакета Fixed-Point задач и эффективности их реализации.

Операции с рабочим пространством средствами пакета Fixed-Point

В пакете Fixed-Point есть средства для операций с рабочим пространством системы MATLAB, то есть записи в него данных и извлечения их Разумеется, речь идет о данных в F-формате. Соответствующие блоки являются цифровыми F-аналогами блоков, имеющихся в разделе Continuous основной библиотеки. В связи с этим ограничимся примером их применения, представленным на рис. 13.24

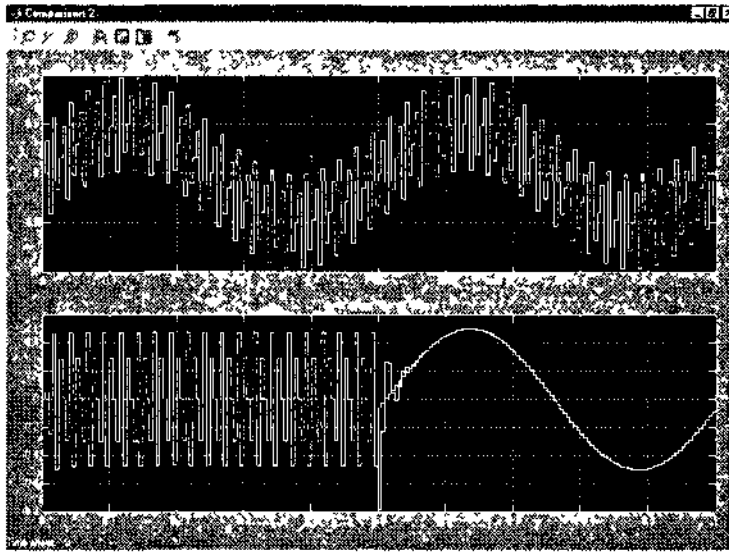


Рис. 13.23. Осциллограммы работы инвариантного по времени цифрового фильтра Баттерворта

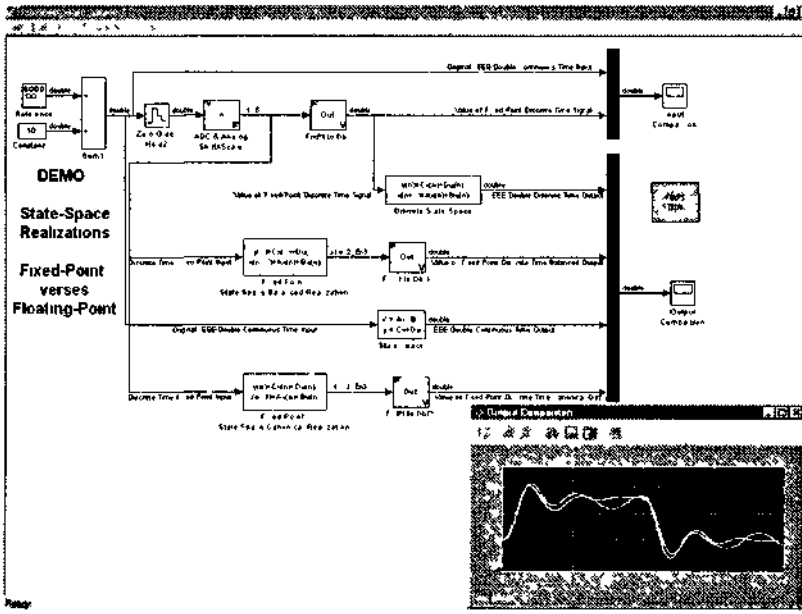


Рис. 13.24. Операции с рабочим пространством средствами пакета Fixed-Point

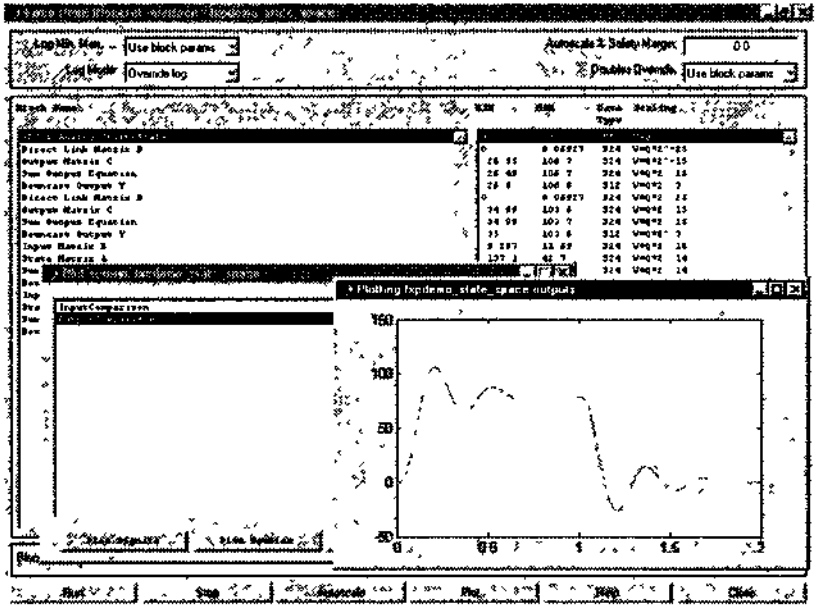


Рис. 13.25. Окно GUI, демонстрирующее операции с рабочим пространством

Этот пример дает наглядное представление о действиях с рабочим пространством. В частности, выполняется сравнение результатов моделирования для обычной реализации модели (без применения F-формата данных) и для реализации с применением F-формата. Осциллограммы показывают, что, несмотря на качественное совпадение результатов, наблюдается количественное расхождение (особенно в представлении колебательных процессов).

На рис. 13.25 показано окно Fixed-Point GUI, иллюстрирующее работу этой модели.

С другими примерами применения пакета Fixed-Point читатель может ознакомиться самостоятельно.

Глава 14. Пакет Digital Signal Processing Blockset

- Обзор пакета Digital Signal Processing (DSP) Blockset 4.0
- Источники и получатели сигналов
- Математические блоки
- Квантование сигналов
- Управление сигналами
- Раздел DSP Signal Attributes
- Переключатели Switches и счетчики Counters
- Раздел Signal Operations
- Раздел DSP Estimation
- Раздел DSP Statistics
- Раздел DSP Filters
- Демонстрационные примеры на применение пакета DSP
- Техника wavelet-преобразований
- Моделирование приемника сигналов точного времени

Обзор пакета Digital Signal Processing (DSP) Blockset

Разделы библиотеки пакета DSP

Пакет Digital Signal Processing Blockset 4.0, или сокращенно DSP, как и пакет Fixed-Point, является расширением Simulink, предназначенным для моделирования разнообразных цифровых устройств. Из-за отсутствия ограничений на формат числовых данных этот пакет является более универсальным, но в отдельных случаях может проигрывать пакету Fixed-Point в скорости выполнения операций.

Изучение этой главы целесообразно только для тех читателей, которые теоретически и практически знакомы с техникой цифровой обработки сигналов. Без этого многие определения и принципы построения моделей будут малопонятными. Задача данной главы — предоставить читателю данные об огромных возможностях пакета DSP в области моделирования устройств цифровой обработки сигналов и проиллюстрировать их примерами практического применения этого мощного инструмента в моделировании цифровых систем и устройств.

Пакеты Fixed-Point и DSP и основная библиотека Simulink имеют много устройств, почти идентичных по назначению. Например, в пакете DSP уже в который раз мы обнаружим знакомые нам источники сигналов (например, константу или генератор синусоидальных сигналов), а также знакомые получатели сигналов (дисплей или осциллограф). Учитывая это обстоятельство, мы сократим описание устройств пакета DSP.

Доступ к средствам пакета DSP из командной строки MATLAB

Для получения информации обо всех средствах пакета DSP (разделах библиотеки и разделах демонстрационных примеров) следует выполнить команду

```
>> help dspblks\dspblks
```

При этом появится список команд для работы с блоками пакета DSP:

```
DSP Blockset
Version 4.0 (R12) 01-Sep-2000

  What's new
    Readme      - New features, bug fixes, and changes in this version
                  To display the Readme file for Version 4.x, type
                  "info dspblks" at the MATLAB command window

  DSP Blockset Support Functions
    dsp_links       - Display and return library link information
    dspfwiz         - Filter realization wizard user interface
    dsplib          - Open DSP Blockset library
    dspstartup      - Default Simulink model settings for DSP systems
    rebuffer_delay - Compute delay introduced by the Buffer block

  DSP Blockset Libraries v4.x

    dspadpt3        - Adaptive filters
```

```
  dsparch3      - Filter structures
  dspbuff3      - Buffers
  dspddes3      - Filter design
  dspfactors    - Matrix factorizations
  dspindex      - Indexing
  dspinverses   - Matrix inverses
  dsplp         - Linear prediction
  dsplibv4      - Main DSP Blockset library for v4 x
  dspmathops    - Math operations
  dspmlti3      - Multirate filters
  dspmtrx3      - Matrix functions
  dspparest3    - Parameter estimation
  dsppolyfun      Polynomial functions
  dspquant2     - Quantizers
  dspsigattribs - Signal attributes
  dspsigops     - Signal operations
  dspsnks3      - DSP sinks
  dspsolvers      Linear system solvers
  dspspect3     - Spectral estimation
  dspsrcs3      - DSP sources
  dspstat3      - Statistics
  dspswit3      - Switches and counters
  dspxfrm3      - Transforms (vector-based)

DSP Blockset Libraries v3 x
  dspadpt2    - Adaptive filter structures
  dsparch2    - Non-adaptive filter architectures
  dspbdsp2    - Basic DSP signal operations
  dspbuff2    - Data buffering blocks
  dspddes2      Digital and analog filter-design blocks
  dspelem2    - Elementary functions library
  dsplibv3    - Main DSP Blockset library for v3 x
  dsplinalg   - Linear algebra library
  dspmlti2    - Multirate filter blocks
  dspmtrx2    - Matrix functions blocks
  dspquant    - Quantizer blocks (added in v3 1)
  dspparest2  - Parameter estimation library
  dspsnks2    - DSP sinks library
  dspspect2   - Spectral analysis library
  dspsrcs2    - DSP sources library
  dspstat2    - Statistics blocks
  dspswit2    - Switches and counters
  dspvect2    - Vector functions operations
  dspxfrm2      Vector signal transforms

DSP Blockset Libraries v2 2 and prior
  dspadpt     - Adaptive filter structures
  dsparch     - Non-adaptive filter architectures
  dspbdsp     - Basic DSP signal operations
  dspbuff     - Data buffering blocks
```

```
dspcmplx   - Complex math blocks
dspddes      Digital and analog filter-design blocks
dspelmat   - Elementary (scalar) math blocks
dsplibv2     Main DSP Blockset library for v2.2
dspmlti      Multirate filter blocks
dspmtrx    - Matrix math operations
dspsnks    - DSP sinks library
dspspect     Spectral analysis library
dspsrcs      DSP sources library
dspstat    - Statistics operations
dspswit    - Switches
dspvect      Vector math operations
dspxfrm      Vector signal transforms
```

Этот список включает в себя разделы библиотек и демонстрационные примеры, как новые (из реализации 12 системы MATLAB + + Simulink), так и сохранившиеся от предшествующих реализаций.

Разделы библиотеки DSP

Доступ к библиотеке DSP, как и к любой библиотеке пакета Simulink, можно получить, набрав имя одного из приведенных выше m-файлов или открыв браузер библиотек. Окно браузера с разделами этой библиотеки показано на рис. 14.1.

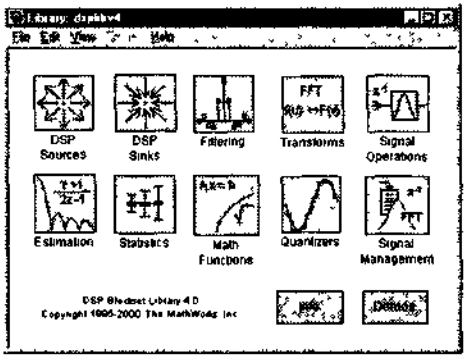


Рис. 14.1. Разделы библиотеки пакета DSP

Библиотека пакета DSP содержит следующие разделы:

- DSP Sources — источники цифровых сигналов DSP;
- DSP Sinks — получатели цифровых сигналов DSP;
- Filtering — средства цифровой фильтрации;
- Transforms — преобразователи информации;

- Signal Operations — средства обработки сигналов;
- Estimation — средства оценки сигналов,
- Statistics — средства статистической обработки сигналов,
- Math Functions — математические функции,
- Quantizers — квантующие блоки,
- Signal Managements — блоки управления сигналами

Есть также пиктограммы доступа к справке по пакету DSP (Info) и открытия стандартного окна MATLAB с демонстрационными примерами (Demos).

Источники и получатели сигналов

Источники сигналов

Окно источников сигналов, представленное на рис. 14.2, содержит 16 блоков. Многие из источников (например, DSP Constant, Random Source, Sine Wave и др.) уже рассматривались, но есть и специфические. Их мы опишем чуть ниже.

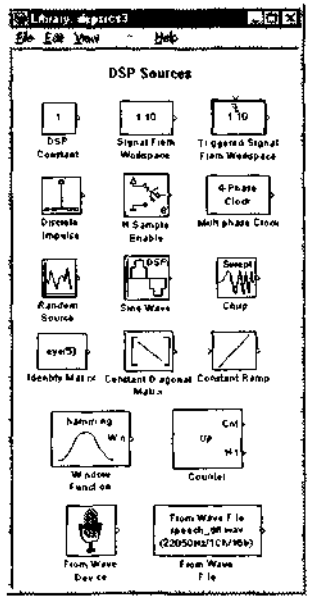


Рис. 14.2. Блоки источников сигналов пакета DSP

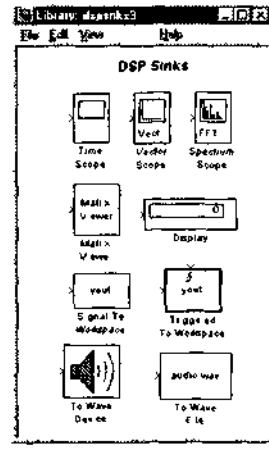


Рис. 14.3. Блоки получателей сигналов

Получатели сигналов

Сигналы обычно контролируются с помощью получателей сигналов, окно с которыми (раздел библиотеки DSP Sinks) представлено на рис. 14.3 Здесь опять видны хорошо знакомые нам устройства, например дисплей Display или осциллограф — он теперь называется Time Scope

Работа с источниками и получателями сигналов

Перенесем в пустое окно модели Simulink источник синусоидального сигнала и осциллограф. Подключив выход источника ко входу осциллографа, получим простейшую модель на основе блоков пакета DSP, представленную на рис. 14.4.

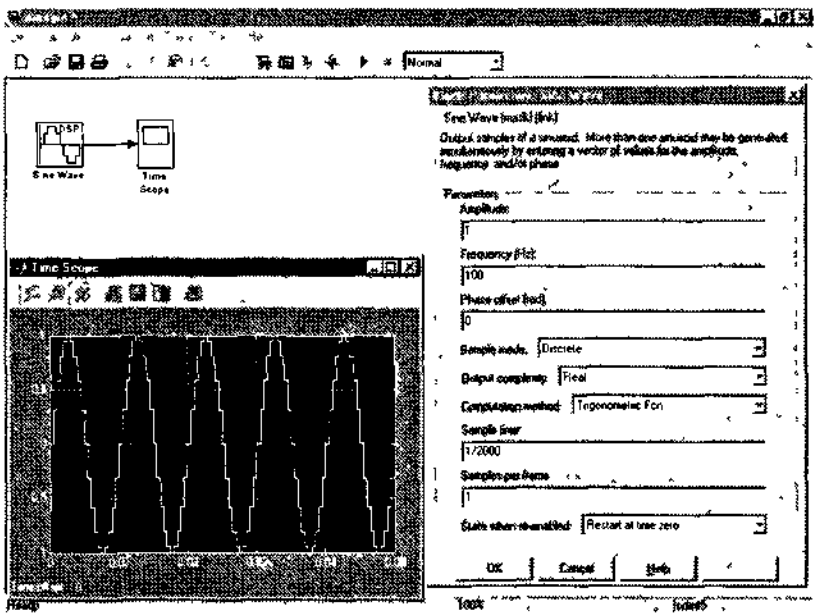


Рис. 14.4. Простейшая модель на основе источников и получателей сигналов пакета DSP

Запустив модель, можно убедиться в том, что источник создает квантованный синусоидальный сигнал, причем квантование имеет равный шаг по времени. Это и фиксирует осциллограф Time Scope, ко-

торый по виду и особенностям применения мало чем отличается от стандартного блока Scope библиотеки Simulink

Окно параметров блока DSP Sine Wave, показанное на рис. 14.4, имеет абсолютно идентичный интерфейс с окном источника Sine Wave стандартной библиотеки Simulink. Правда, список параметров источника выглядит более внушительным. В нем появился ряд новых, но достаточно очевидных параметров: режим дискретизации сигнала Sample mode, тип выходного сигнала (действительный или комплексный) Output Complexity, метод вычислений Computation method, период фрейма Samples per frame (в тактах эталонного времени) и установка статуса при повторном исполнении блока State when re-enabled.

Рисунок 14.5 показывает применение наиболее распространенных источников сигналов и их получателей из пакета DSP. Здесь дано сразу несколько примеров, что возможно вследствие параллельной работы ряда моделей в одном окне. Однако в общем случае может возникнуть проблема синхронизации работы блоков, поскольку все модели будут работать с одинаковым шагом модельного времени в одном и том же интервале его изменения

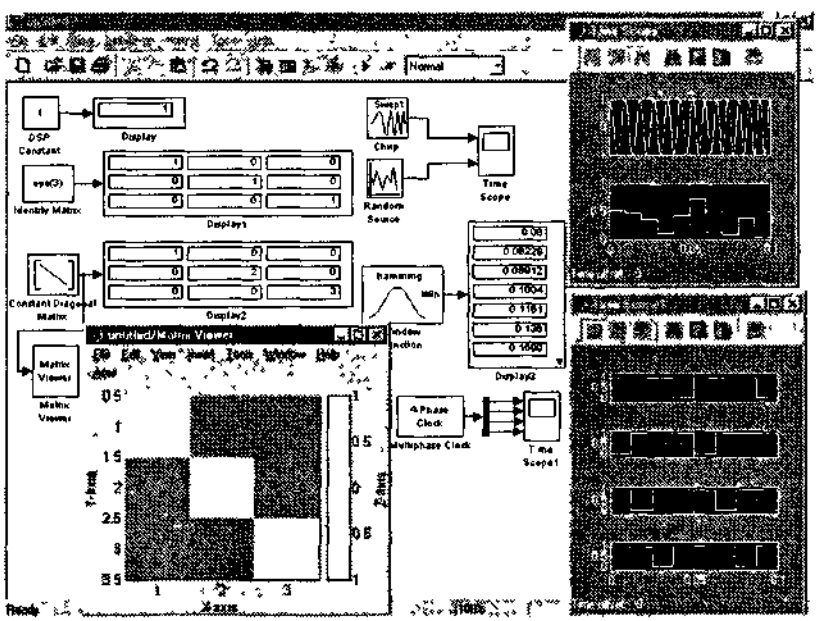


Рис. 14.5. Примеры работы с источниками и получателями сигналов

Из этих примеров можно увидеть работу некоторых новых блоков пакета DSP. Например, это блоки формирования единичной матрицы Identity Matrix, диагональной матрицы Constant Diagonal Matrix и просмотра структуры матрицы Matrix Viewer. Квантованные сигналы в виде синусоиды с линейно нарастающей частотой и в виде шума дают блоки Chirp и Random Source. К новым блокам относится многотактный генератор прямоугольных импульсов. Все эти примеры наглядно свидетельствуют о едином подходе к применению источников сигналов и их получателей. Специальные типы виртуальных регистрирующих устройств пакета DSP мы рассмотрим далее.

Математические блоки

Раздел библиотеки Math Function

Рассмотрим блоки пакета DSP, выполняющие математические операции. Именно наличие таких операций делает модели пакета математически и физически прозрачными и наглядно показывает роль математических вычислений при моделировании сложных систем и устройств, к которым в полной мере относятся современные цифровые системы связи.

Пакет DSP имеет весьма обширные возможности в проведении математических операций. Они сосредоточены в разделе библиотеки с названием Math Function. Структура этого раздела видна на рис. 14 6, на котором показано окно браузера библиотек Simulink с открытым разделом математических функций.

Работа с блоками математических операций

Один из подразделов — Math Operations — показан на рис. 14.6 в открытом виде. Этот раздел содержит блоки вычисления exp(jx), вычисления суммы с накоплением, преобразования децибел, вычисления разности элементов в столбцах матриц и нормализации элементов матрицы. Все это довольно очевидные операции, и мы проиллюстрируем их примерами, представленными в левой части рис. 14.7.

Обращение матриц

Пакет DSP имеет четыре блока обращения матриц. Окно с этими блоками представлено на рис. 14.8, а примеры применения блоков даны на рис. 14.7 в правом верхнем углу. Обратите внимание, что

смысл операции обращения в разных методах разный, а потому отличаются и результаты.

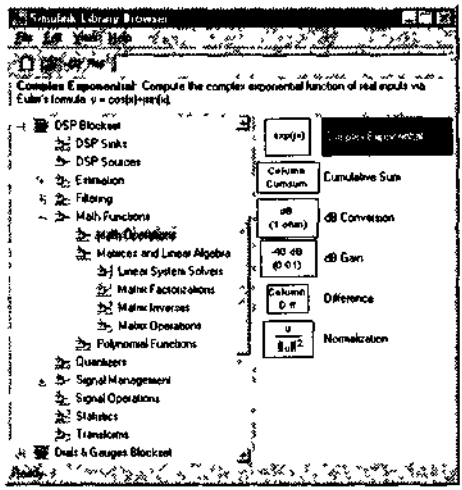


Рис. 14.6. Раздел Math Functions библиотеки пакета DSP

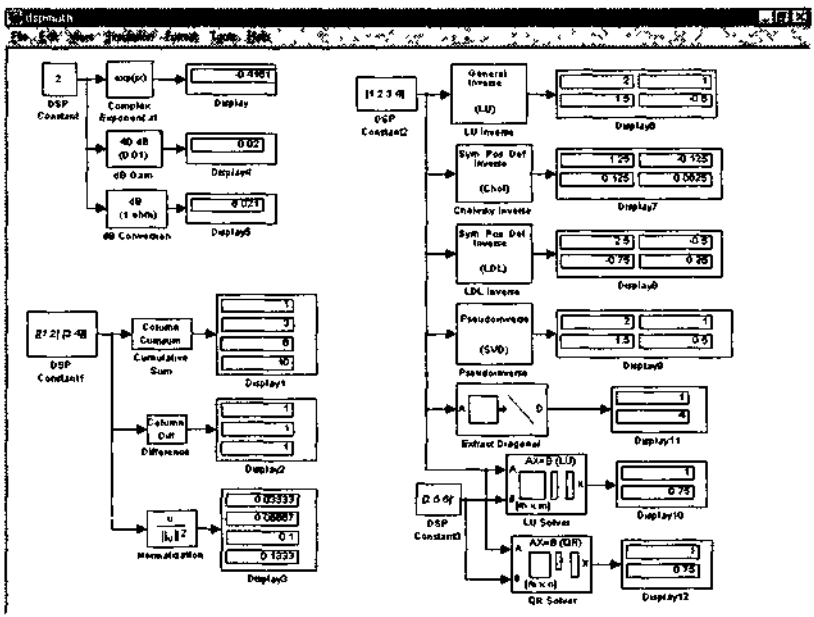


Рис. 14.7. Примеры работы с блоками математических операций

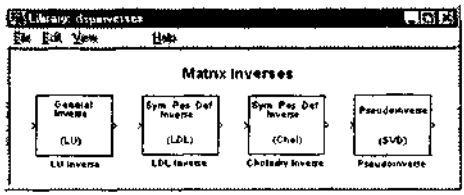


Рис. 14.8. Блоки обращения матриц

Типовые матричные операции

Другие матричные операции представлены весьма внушительным набором блоков (рис. 14.9). Ввиду их общеизвестности комментировать эту группу операций мы не будем — названия блоков и их пиктограммы говорят сами за себя В качестве примера в правой части рис. 14.7 показано выделение диагонали матрицы с помощью блока Extract Diagonal

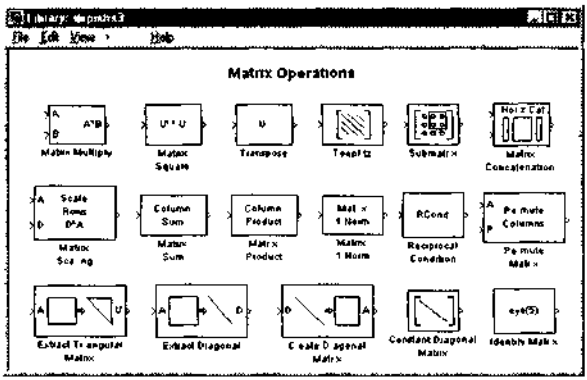


Рис. 14.9. Окно с блоками типовых матричных операций

Ряд примеров на применение большинства блоков этого раздела представлен на рис. 14 10 Обратите внимание, что большинство операций выполняется над простой квадратной матрицей [1 2; 3 4] Для регистрации результатов операций использован цифровой индикатор — дисплей, пиктограмма которого растягивается до размера, достаточного для наблюдения всех результатов.

Решение систем линейных уравнений

Важной задачей линейной алгебры является решение систем линейных уравнений вида $AX = B$, где A — квадратная матрица коэффи-

циентов правой части уравнения и B — вектор-столбец свободных членов (правая часть системы линейных уравнений). Хотя решение такой системы возможно с помощью матричных блоков (пример дан на рис. 14.10 в верхнем правом углу окна), пакет DSP предоставляет 4 специальных блока для решения систем линейных уравнений (рис. 14.11)

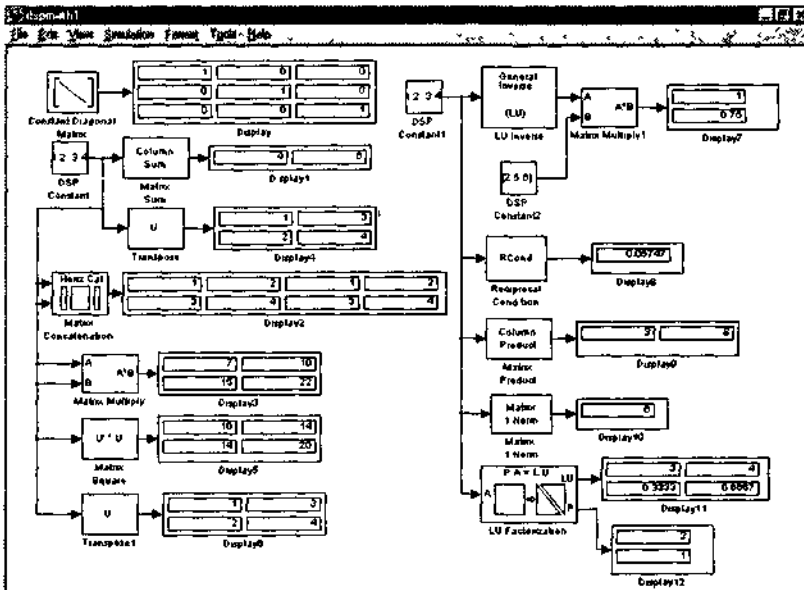


Рис. 14.10. Примеры применения блоков типовых матричных операций

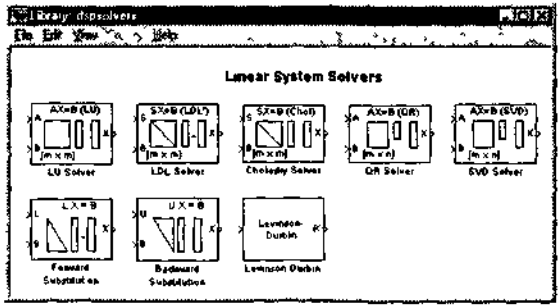


Рис. 14.11. Окно с блоками решения систем линейных уравнений

В правой части рис 14 7 даны два примера на решение системы линейных уравнений

$x_1 + 2x = 5$
$3x_1 + 4x = 6$

Обратите внимание в этих примерах на форму задания матрицы A и вектора B Они заданы как [1 2, 3 4] и [2 5 6]'. Здесь апостроф означает транспонирование вектора, то есть превращение его из вектора строки в вектор-столбец.

Факторизация матриц

Факторизация матриц в пакете DSP реализована пятью блоками (рис 14 12)

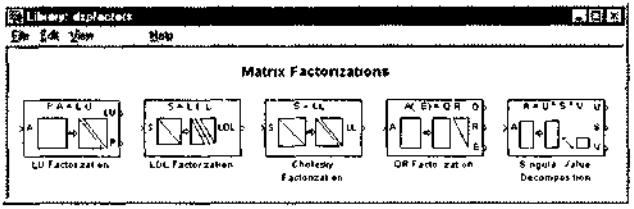


Рис. 14.12. Блоки факторизации матриц

Их пиктограммы явно указывают, что должно быть на входе блоков и что получается на выходе. Пример LU-разложения дан на рис. 14 10

Операции с полиномами

Полиномом называют степенной многочлен с целочисленными показателями степени. В Simulink он задается значением независимой переменной x и вектором коэффициентов, расположенных в порядке убывания степени Обязательно следует указывать даже нулевые коэффициенты Например, вектор коэффициентов [3 2 1 –5] задает полином $3u^3 + 2u^2 + u - 5$, а вектор [2 0 3] — полином $-2u^2 + 3$ (степень первого порядка отсутствует)

Для работы с полиномами пакет DSP предлагает всего три блока (рис 14.13) Эти блоки обеспечивают вычисление значений полинома, уточнение его коэффициентов по минимуму среднеквадратической погрешности в заданных точках и определение устойчивости полинома по его корням

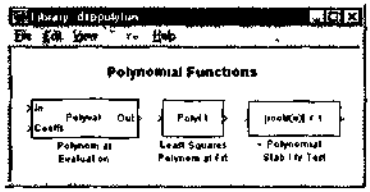


Рис. 14.13. Блоки операций с полиномами

Примеры применения этих блоков представлены на рис. 14 14 Обратите внимание на блок вычисления значения полинома Polynomial Evalution. Он использован трижды (при вычислении значения полинома для элементов входного вектора, для построения графика значений полинома при синусоидальном сигнале на входе и для уточнения коэффициентов полинома). Окно параметров блока регрессии дано в левом нижнем углу окна рис. 14.14.

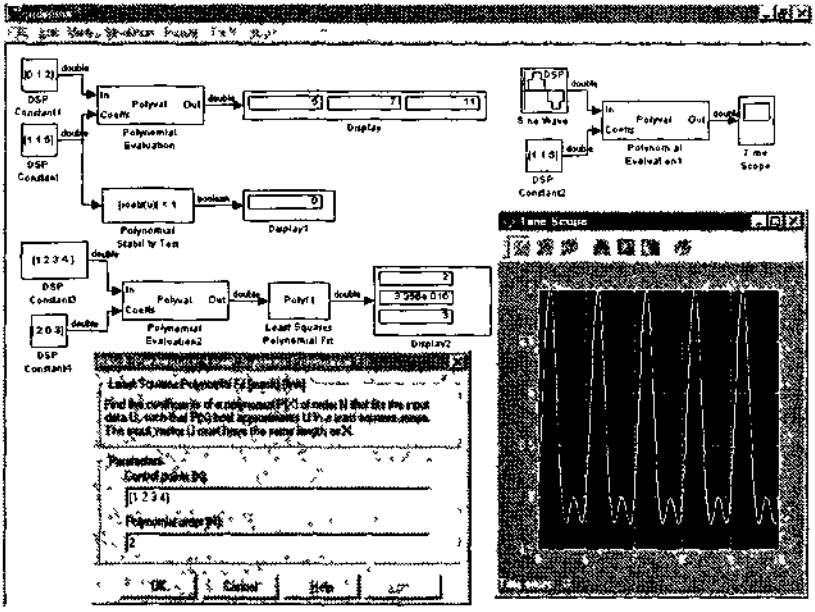


Рис. 14.14. Операции с полиномами

Квантование сигналов

В состав раздела Quantizers (Квантователи) входят три блока, представленные в окне рис. 14.15. Это блок собственно квантователя

Quantizer, кодирующий блок Uniform Encoder и декодирующий блок Uniform Decoder

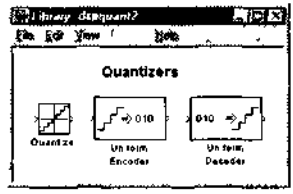


Рис. 14.15. Блоки квантования

На рис. 14.16 дана простая и наглядная модель канала для цифрового кодирования и декодирования «синусоидального» сигнала с нарастающей во времени частотой.

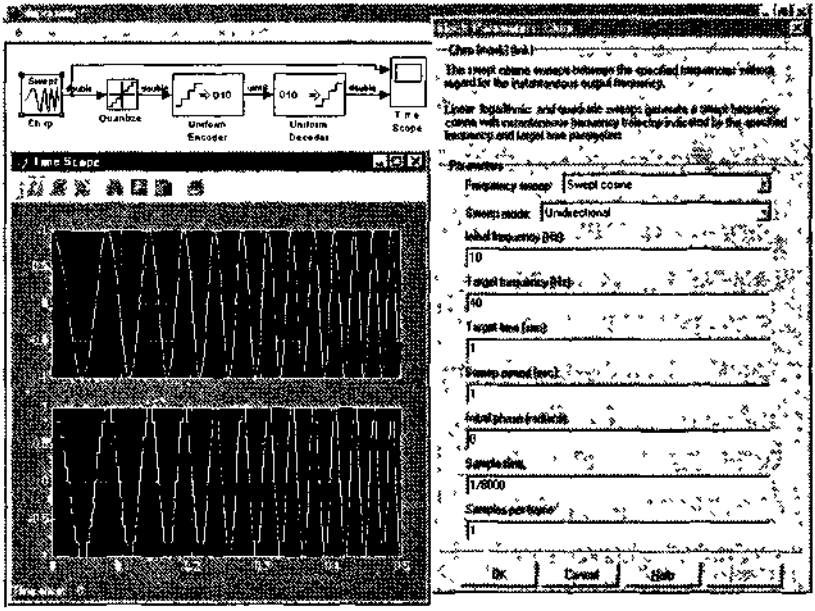


Рис. 14.16. Пример квантования сигнала переменной частоты

Слово «синусоидальный» взято в кавычки потому, что такой сигнал строго теоретически уже не является синусоидальным. Этот сигнал, созданный свип-генератором Chirp, подается на квантователь и затем с помощью кодирующего устройства превращается в последовательность цифровых кодов. Поток кодов поступает на вход де-

кодирующего устройства и превращается в квантованный исходный сигнал. На практике для получения выходного сигнала без заметных ступенек применяют специальные фильтры (об их проектировании и моделировании речь пойдет ниже).

Окно параметров свип-генератора показано на рис. 14 16 справа. Окна параметров кодирующего и декодирующего блока просты и потому не показаны Однако для получения представленного на рис. 14.16 результата в них следует установить следующие параметры блоков (часть из них отлична от принятых по умолчанию).

Параметр блока	Uniform Coder	Uniform Decoder
Амплитуда Peak	1	1
Разрядность Bits	8	8
Режим переполнения Overflow mode	Параметра нет	Saturate
Тип выходного сигнала Output type	Unsigned integer	Double

Управление сигналами

Обзор раздела Signal Managements

Средства управления сигналами сосредоточены в разделе Signal Managements библиотеки пакета DSP (рис. 14.17)

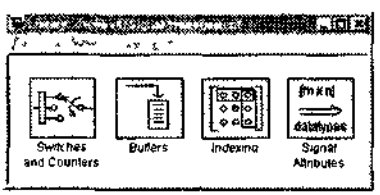


Рис. 14.17. Раздел управления сигналами

Этот раздел содержит четыре подраздела.

- Switches and Counters — блоки переключения сигналов и счетчики,
- Buffers — блоки буферизации сигналов и временной задержки;
- Indexing — блоки индексирования,
- Signal Attributes — блоки атрибутов сигнала

Блоки подраздела Buffers

Блоки подраздела Buffers представлены на рис. 14.18.

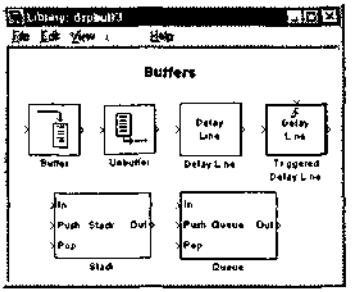


Рис. 14.18. Блоки подраздела Buffers

Работа блока Buffer

Блок Buffer служит для буферизации сигналов. Его работу можно уподобить получению воды из единственного крана с помощью ведер — заполняется одно ведро, затем другое и т. д. Таким образом, поток данных сигнала дробится на части (фреймы) заданного размера (рис. 14.19).

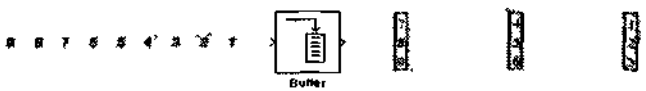


Рис. 14.19. Работа блока Buffer

Буфер характеризуется тремя параметрами (в скобках дано значение параметра по умолчанию):

- Buffer size — размер буфера ($M = 64$), то есть количество последовательных значений сигнала, образующих фрейм;
- Buffer overlap — перекрытие, то есть число элементов предыдущего фрейма, повторяющихся в последующих фреймах ($L = 0$);
- Initial condition — начальное состояние при tm = 0 (0).

Входным сигналом может быть не только последовательность одиночных значений, но и векторы и матрицы. Рисунок 14.20 иллюстрирует работу буфера при входном сигнале в виде вектора.

Рисунок 14.21 показывает работу буфера при $M = 3$ и $L = 1$, то есть при наличии перекрытия.

Рис. 14.20. Работа блока Buffer при векторном входном сигнале

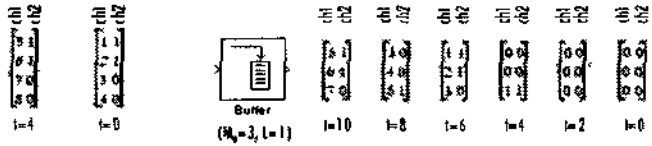


Рис. 14.21. Работа блока Buffer при наличии перекрытия

Если входной вектор содержит N элементов, а параметр Buffer size равен M, то, как видно из рис. 14.20, каждый фрейм будет матрицей размером $M \times N$, каждый элемент которой скаляр, полученный по отдельному каналу h_i (i — номер канала).

Если входной сигнал — матрица размером $m \times n$, то она преобразуется в вектор, содержащий mn элементов.

Работа блока Unbuffer

Блок Unbuffer собирает фреймы в один поток (рис. 14.22). Если фреймы разделены интервалом времени 1, то скалярные значения выходного сигнала будут идти с интервалом t/M (в нашем примере 1/3).

Рис. 14.22. Работа блока Unbuffer при векторном входном сигнале

Этот блок имеет один параметр — начальное состояние Initial condition (0). Рисунок 14.23 показывает работу блока при четырехка-

нальном входном сигнале, каждый фрейм которого — матрица размером 3 × 4.

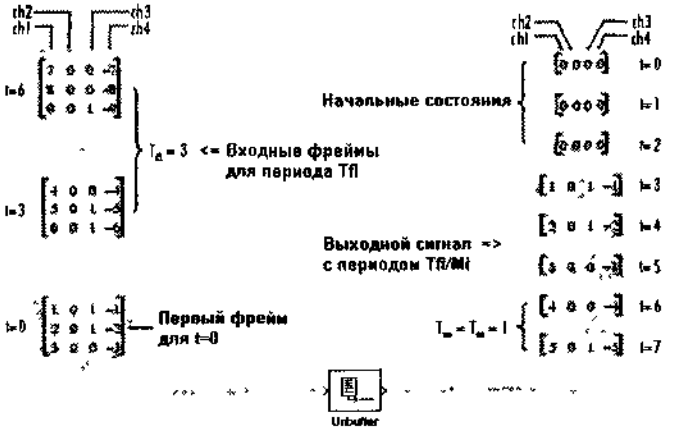


Рис. 14.23. Работа блока Unbuffer при матричном входном сигнале

Организация очереди (блок Queue)

Блок Queue (Очередь) служит для организации типа данных, называемого очередью или FIFO-регистром (от слов First In — First Out, что означает «Первым вошел — первым вышел»). Механизм очереди поясняет рис. 14.24.

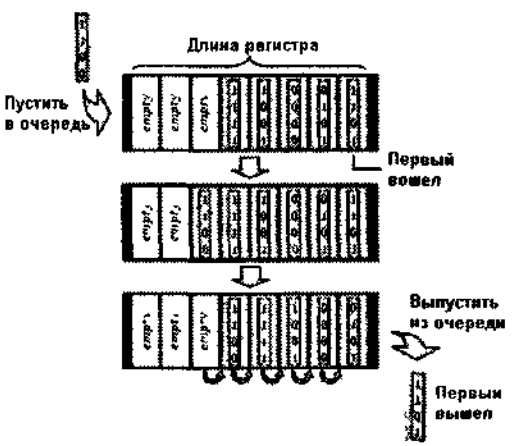


Рис. 14.24. Организация очереди с помощью блока Queue

Если на порт Push дан сигнал «Пустить в очередь», то очередное значение (вектор или матрицу) сигнала со входного порта In помещается в конец очереди. А если на порт Pop подан сигнал «Выпустить из очереди», то значение сигнала (вектор или матрица), помещенное в очередь первым, будем выпущено из очереди первым. Есть еще порт Clr, сигнал на котором очищает очередь. Если события (сигналы) поступают одновременно на все порты, то сначала выполняется очистка очереди (Clr), затем пуск в нее (Push) и, наконец, команда вывода значения из очереди Pop.

Окно параметров блока Queue показано на рис. 14.25.

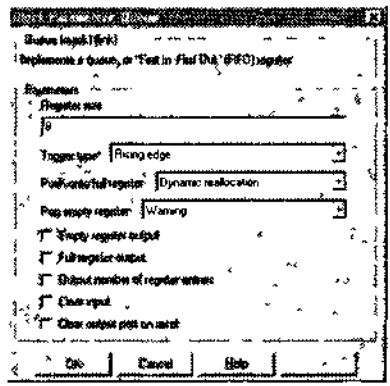


Рис. 14.25. Работа блока Queue

Блок Queue имеет следующие параметры:

- Register size — размер регистра, или максимальная длина очереди;
- Trigger type — способ запуска: по подъему сигнала (Rising edge), спаду (Falling edge) и по любому изменению сигнала (Either edge);
- Push onto full register — действия при заполнении регистра: Ignore — игнорировать, Warning — выдать предупреждение, Error — вызвать сообщение об ошибке и Dynamic rellocations — динамически изменить размер регистра;
- Pop empty register — действия при пустом регистре (см. выше, за исключением динамического изменения размера регистра).

Кроме того, в окне имеется пять флажков:

- Empty redister output — задает возможность использования выходного порта Empty (сигнал 1 на его выходе означает, что регистр пуст, а 0 — что в нем есть данные);

- Full register output — возможность использования выходного порта Full (сигнал 1 на его выходе означает, что регистр заполнен, 0 — что он пуст);
- Output number of register entieres — управляет использованием выходного порта, сигнал которого указывает, сколько элементов может принять регистр в данный момент времени;
- Clear input — возможность использования входного порта Clear для очистки очереди (регистра);
- Clear output port on reset — возможность задания нуля на выходе очереди, если поступил сигнал очистки очереди Clr.

Организация стека (блок Stack)

Стек — это форма организации данных типа LIFO (Last Input, First Out — «Последним пришел — первым ушел»). Стек можно уподобить стопке тарелок — ту, которую положили последней, можно взять первой. Блок Stack при наличии сигнала на входе Push помещает значение (вектор или матрицу) в свою вершину (остальные данные смещаются вниз) При наличии сигнала на входе Pop данные, находящиеся в вершине стека, выносятся из стека. При подаче сигнала на вход Clr стек очищается. Диаграмма работы стека показана на рис. 14.26.

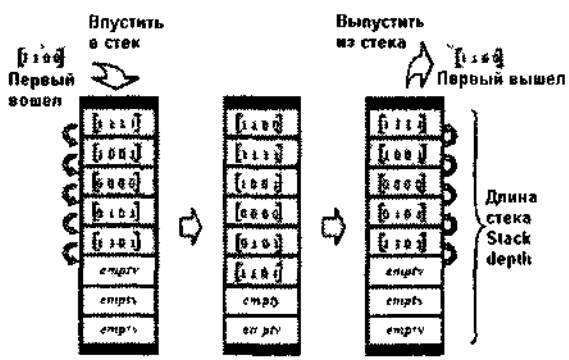


Рис. 14.26. Диаграмма работы блока Stack

Окно параметров блока стека имеет такой же вид и те же параметры, что и окно параметров блока Queue, только вместо параметра Register size имеется параметр Stack depth (глубина стека).

Организация сдвигового регистра (блок Delay Line)

Для организации сдвигового регистра в предшествующей версии Simulink предназначался блок Shift Register. В Simulink 4.0 он переименован в Delay Line. Диаграмма работы блока представлена на рис. 14.27.

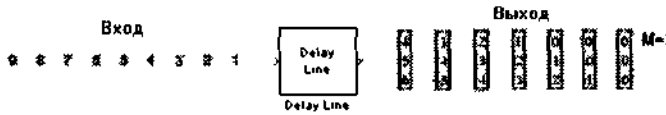


Рис. 14.27. Работа блока Delay Line

Этот блок имеет два установочных параметра: Delay line size — размер выходного фрейма и Initial condition — значение сигнала на выходе на начальном шаге (tm = 0).

Блок Triggered Delay Line

Блок Triggered Delay Line имеет дополнительный управляющий вход. Подача определенного перепада на управляющий вход переводит регистр в режим работы обычного сдвигового регистра, подача противоположного перепада блокирует работу регистра. Если на управляющий вход поступает последовательность импульсов, сдвигающий регистр поочередно то работает, то нет. Таким образом, функционально он подобен сдвигающему регистру с триггером на входе.

Окно параметров блока Triggered Delay Line показано на рис. 14.28. Помимо указанных для блока Delay Line параметров блок Triggered Delay Line имеет параметр Trigger type, определяющий тип управляющего сигнала: можно задать переключение режима работы регистра при положительном, отрицательном или при любом перепаде.

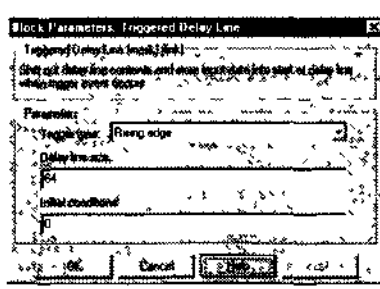


Рис. 14.28. Окно параметров блока Triggered Delay Line

Подраздел DSP Signal Attributes

Подраздел Signal Attributes раздела Signal Managements библиотеки пакета DSP представлен окном, показанным на рис. 14.29.

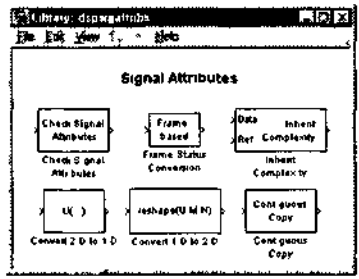


Рис. 14.29. Блоки подраздела Signal Attributes

В этот подраздел входит шесть блоков:

- Convert 2-D to 1-D — преобразование двумерного сигнала в одномерный;
- Convert 1-D to 2-D — преобразование одномерного сигнала в двумерный;
- Continuous Copy — создает продолженную копию данных;
- Check Signal Attributes — проверка атрибутов сигнала;
- Frame Status conversion — преобразование статуса фреймов;
- Inherit Complexity — наследование комплексности данных.

На рис. 14.30 даны примеры работы этих блоков.

Из этих блоков лишь три первых блока применяются достаточно широко. Блоки Continuous Copy и Convert 2-D to 1-D не имеют параметров. Блок Convert 1-D to 2-D имеет параметры, задающие число строк и столбцов двумерного сигнала.

Переключатели Switches и счетчики Counters

Подраздел Switches and Counters раздела Signal Managements представлен блоками переключателей и счетчиков (рис. 14.31).

В этом подразделе библиотеки имеется шесть блоков:

- N-Sample Enable — блок включения неактивного сигнала; задает в течение заданного времени неактивное состояние сигнала, а затем — активное;

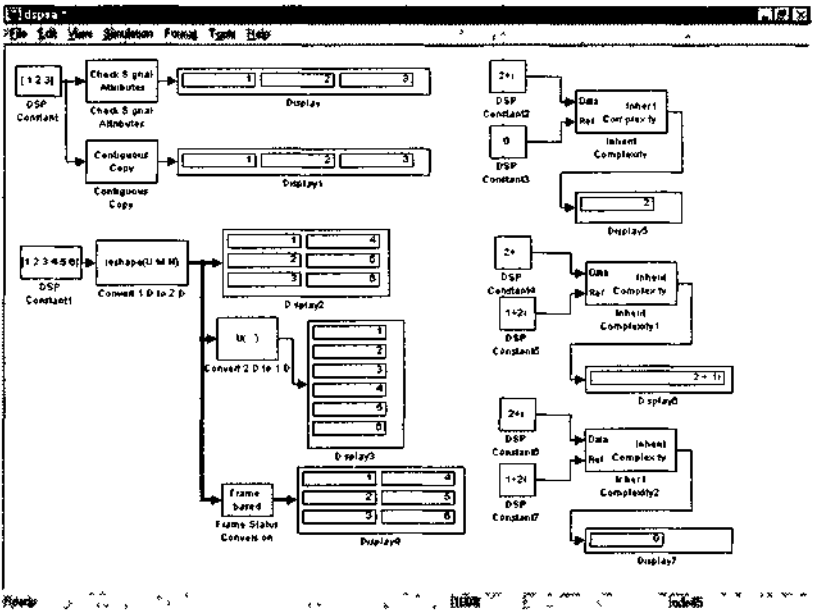


Рис. 14.30. Примеры работы блоков Signal Attributes

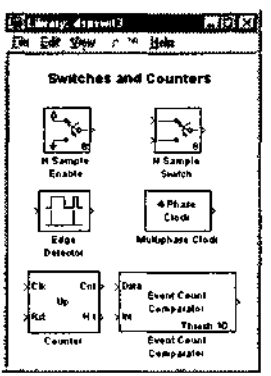


Рис. 14.31. Блоки подраздела Switches and Counters

- N-Sample Switch — блок переключения неактивного сигнала; задает в течение заданного времени состояние выходного сигнала, равное уровню сигнала на одном входе, а затем — уровню сигнала на другом входе;
- Edge Detector — детектор перехода через ноль, который создает выходной импульс с единичной амплитудой и длительностью при переходе входного сигнала через 0;

- Multiphase clock — блок многофазных импульсов;
- Counter — счетчик;
- Event-Count Comparator — блок подсчета событий.

Переключающие блоки

Работу блоков N-Sample Enable, N-Sample Switch и Edge Detector поясняет рис. 14.32. Он содержит осциллограммы работы блоков и окна их параметров. Параметр N задает число тактов, по истечении которых меняется состояние переключателей.

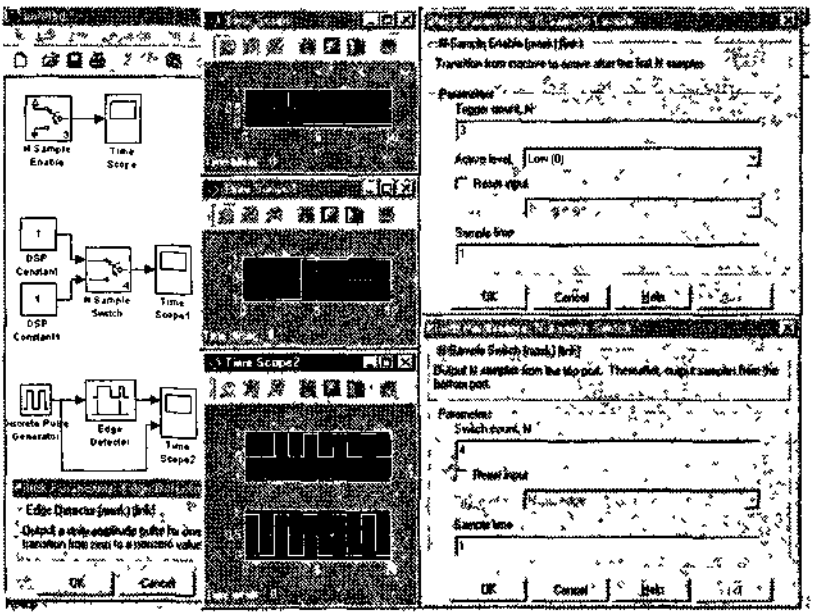


Рис. 14.32. Работа переключателей подраздела Switches and Counters

Окно параметров блока N-Sample Enable позволяет установить следующие параметры.

- Trigger count — интервал запуска (число тактов эталонного времени), в течение которого блок выдает неактивный уровень выходного сигнала (противоположный значению Active level);
- Active level — активный уровень: высокий High (1) или низкий Low (0);
- Trigger type — условие возвращения блока в исходное состояние: Rising edge — при подъеме входного сигнала, Falling edge — при спаде и Either edge — в любом случае;

- Sample time — эталонное время.

Кроме того, можно установить флажок Reset input, включающий вход сброса, сигнал на котором сбрасывает блок в исходное состояние. Блок N-Sample Switch имеет те же параметры, за исключением того, что первый параметр назван Swith count.

Пример применения многофазного генератора Multiphase clock приведен на рис. 14.5 в правом нижнем углу.

Счетчики

Работу счетных блоков Counter и Event-Count Comparator поясняет рис. 14.33. На нем также показаны окна параметров этих блоков.

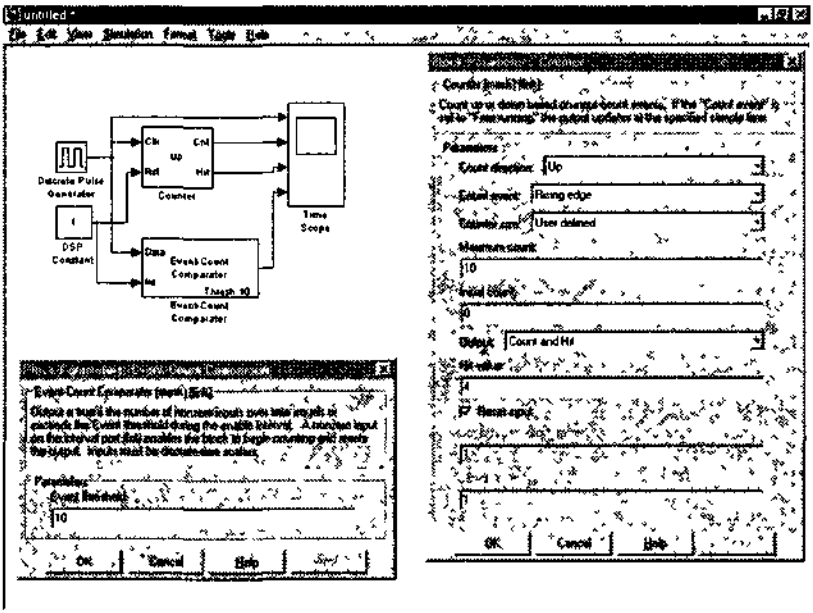


Рис. 14.33. Блоки счетчиков подраздела Switches and Counters

Блок Event-Count Comparator подсчитывает число ненулевых значений входного сигнала, поступивших на вход Data в течение интервала времени, заданного сигналом на управляющем входе Int. Когда это число достигает порога Event threshold, блок выдает импульс единичной амплитуды и единичной длительности. Окно параметров блока Event-Count Comparator имеет единственный параметр — уже упомянутый порог Event threshold.

Блок Counter подсчитывает число импульсов на входе Clk и меняет значение внутреннего счетчика. Оно контролируется выходом Cnt. Имеется также выход Hit, выдающий импульс единичной амплитуды и длительности, если значение сигнала на выходе Cnt достигнет заданной величины.

Окно параметров блока Counter позволяет установить следующие параметры:

- Count direction — направление счета (Down — в сторону убывания, Up — в сторону возрастания);
- Count event — тип события, влияющего на значения счетчика: Rising edge — подъем входного сигнала, Falling edge — спад, Either edge — любой перепад входного сигнала, Nonzero sample — появление любого ненулевого значения и Free running — отключение порта Clk;
- Counter size — разрядность счетчиков: 8, 16 и 32 бит или User defined — задаваемая пользователем разрядность;
- Maximum count — максимальная разрядность счетчика (если выбрано значение User defined параметра Counter size);
- Initial count — установка начального состояния счетчика;
- Output — выбор конфигурации выходных портов (Count, Hit или оба вместе);
- Samples per output frame — число импульсов в выходном фрейме;
- Sample time — эталонное время.

Имеется также флажок Reset input — разрешение использования порта сброса Rst.

Осциллограммы работы этих блоков представлены на рис. 14.34.

Обработка сигналов (раздел Signal Operations)

Обзор раздела Signal Operations

Раздел Signal Operations библиотеки пакета DSP представлен двенадцатью блоками, показанными на рис 14.35. Эти блоки осуществляют различные операции с сигналами, такие как изменение числа отсчетов сигналов, повторение сигнала, задержки различного вида, свертки и др. Мы рассмотрим лишь основные из этих блоков.

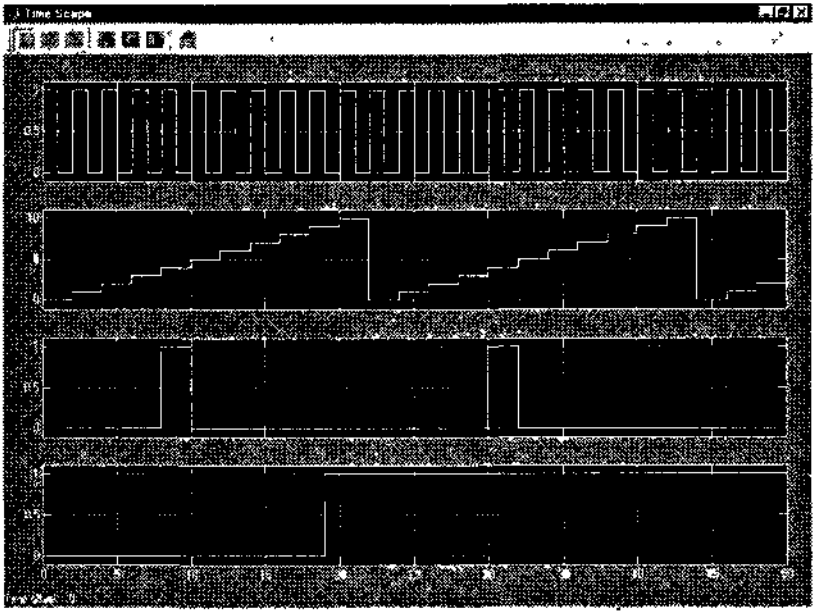


Рис. 14.34. Осциллограммы счетчиков

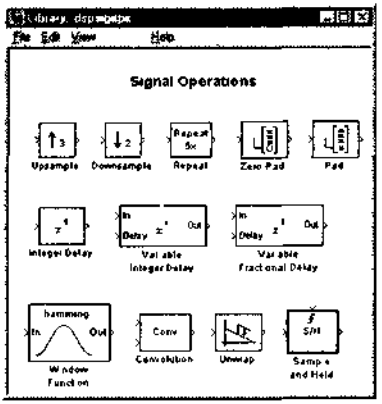


Рис. 14.35. Окно раздела Signal Operations библиотеки DSP

Блок свертки Convolution

Блок Convolution осуществляет операцию свертки для двух векторов. Пример применения этого блока дан на рис. 14.36. Параметров этот блок не имеет.

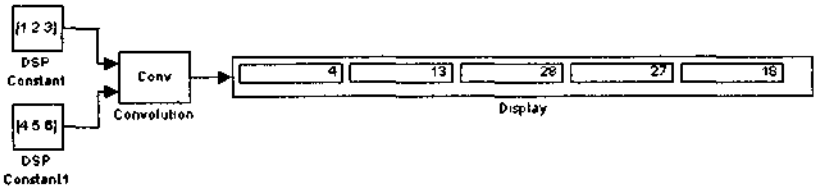


Рис. 14.36. Пример свертки двух векторов

Другие блоки раздела Signal Operations

Функции ряда блоков этого раздела хорошо иллюстрируются временными диаграммами их работы, представленными на рис. 14.37.

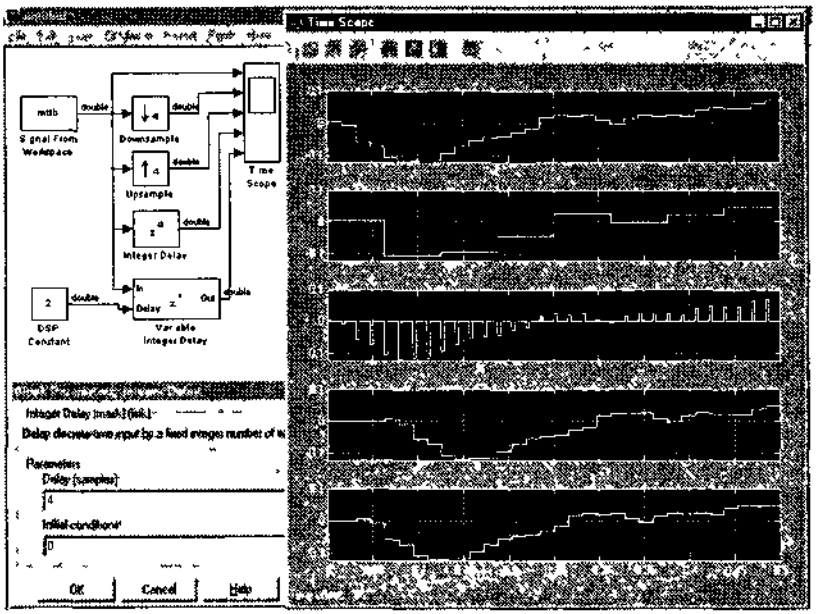


Рис. 14.37. Примеры работы некоторых блоков раздела Signal Operations

Из этих блоков, пожалуй, наиболее важными являются блоки временной задержки сигналов Integer Delay (фиксированная временная задержка) и Variable Integer Delay (управляемая сигналом временная задержка). Для повторения сигнала на заданном промежутке времени служит блок Repeat. С другими, довольно редкими в применении блоками читатель может ознакомиться самостоятельно, используя справочную систему пакета DSP

Раздел DSP Estimation

Обзор раздела Estimation

Раздел библиотеки Estimation (блоки оценки) представлен основным разделом и тремя подразделами (рис.14.38).

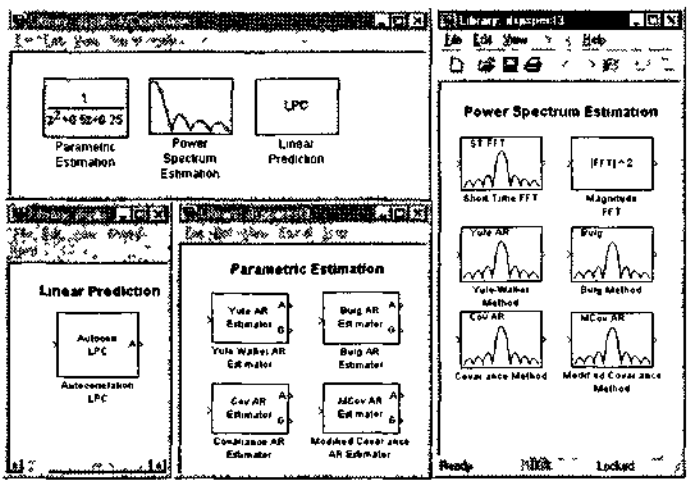


Рис. 14.38. Окно раздела Estimation

Блок автокорреляции Autocorrelation LPT

Окно подраздела линейного предсказания Linear Prediction содержит единственный блок Autocorrelation LPT. Он вычисляет параметры автокорреляции для матрицы (в общем случае $M \times N$) или вектора. Его применение поясняет рис. 14.39. Там же показано и окно параметров этого блока.

Блок возвращает вектор коэффициентов A предсказывающего полинома порядка N, минимизирующего данные по критерию среднеквадратической ошибки по методу LPT. Кроме того (если это задано параметром Output(s)), он возвращает коэффициенты так называемого рефлексного полинома K порядка $N - 1$. Флажок Output prediction error power позволяет задать вывод мощности ошибки P Флажок Inherit prediction order from input dimensions задает наследование порядка предсказывающего полинома от данных размерности входного сигнала.

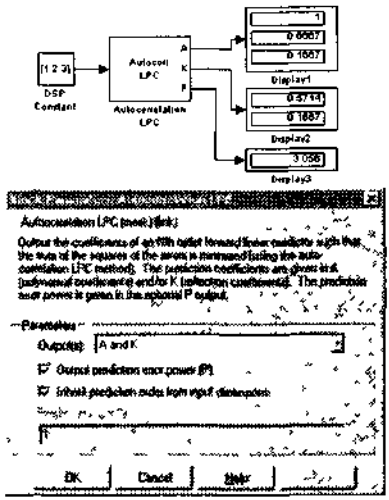


Рис. 14.39. Пример применения блока Autocorrelation LPT

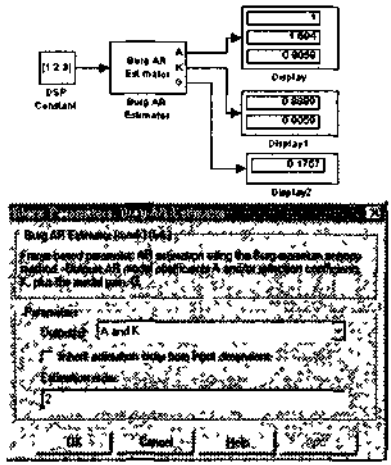


Рис. 14.40. Пример включения блока Burg AR Estimation

Блоки параметрической оценки

Блоки параметрической оценки представлены четырьмя типами: Yule Walker AR Estimator, Burg AR Estimation, Covariabce AR Estimation и Modified Covariance AR Estimation.

Эти блоки обеспечивают оценку, реализуемую различными методами, указанными в их названиях. Поскольку они включаются и используются практически одинаково, ограничимся примером подключения одного блока — Burg AR Estimation, представленным на рис. 14.40. В окне параметров задаются ранее описанные (в предшествующем разделе) параметры.

Раздел оценки спектра мощности Power Spectrum Estimation содержит шесть блоков анализаторов спектра, основанных на быстром преобразовании Фурье (FFT) и методах, реализованных в блоках оценки, упомянутых выше. Пример применения блоков анализаторов спектра представлен на рис. 14.41.

К важнейшим параметрам этих блоков относятся длина (число полос) спектрограммы и параметр наследования размерности входных данных.

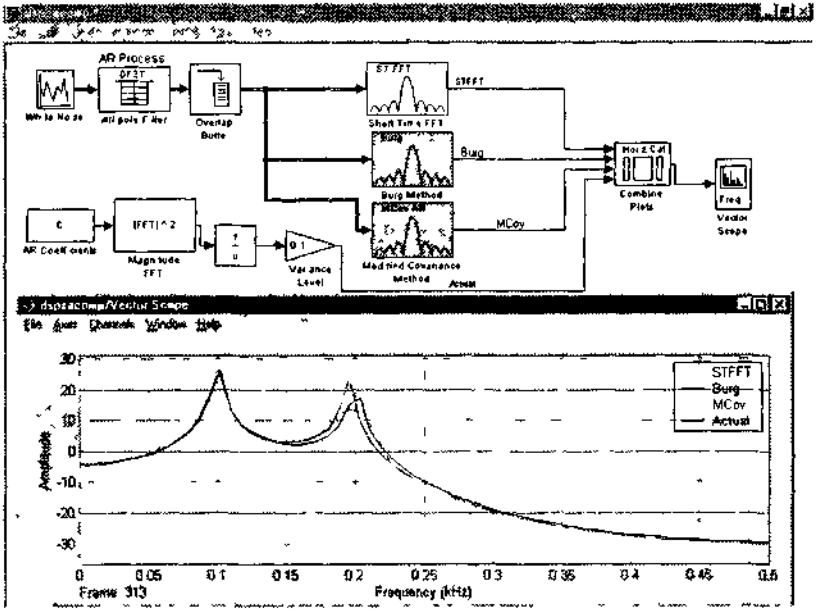


Рис. 14.41. Сравнение анализаторов спектра разного типа

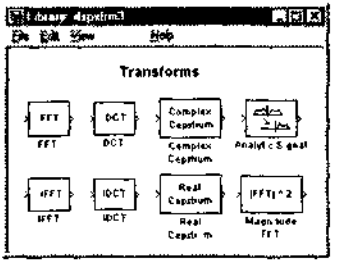


Рис. 14.42. Блоки раздела Transforms

Преобразования сигналов (раздел Transforms)

Обзор раздела Transforms

Раздел Transforms содержит 8 блоков (рис. 14 42).

В этом разделе представлены блоки прямого и обратного преобразования Фурье (FFT) и дискретного косинусного преобразования (DCT), а также блоки связанных с ними операций.

Прямое и обратное преобразования Фурье

Блоки FFT и IFFT реализуют стандартное прямое и обратное преобразования Фурье для входной последовательности данных, число элементов которой должно быть целой степенью числа 2.

Рисунок 14.43 представляет простой пример таких преобразований.

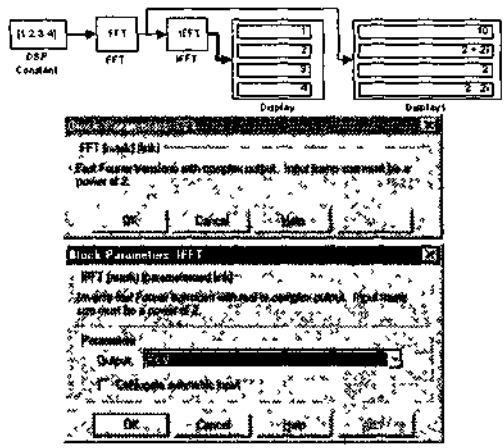


Рис. 14.43. Пример преобразований FFT и IFFT

Здесь вектор из четырех (2^2) чисел [1 2 3 4] вначале подвергается прямому быстрому преобразованию Фурье (блок FFT), а затем обратному быстрому преобразованию Фурье (блок IFFT). В первом случае получается вектор с комплексными числами. Они представляют амплитуды и фазы гармоник входного сигнала (вектора). Таким образом, временное представление сигнала переводится в частотное. Во втором случае частотное представление сигнала переводится во временное. Как нетрудно заметить, в данном случае произошло восстановление исходного сигнала после преобразований FFT и IFFT.

Окно параметров блока FFT является чисто информационным и параметров не содержит. Окно параметров блока IFFT содержит единственный параметр — тип выходного сигнала Output (Real — вещественный выходной сигнал, Complex — комплексный). Возможна также установка флажка Conjugent symmetric input (установка симметричного комплексно-сопряженного входа).

Прямое и обратное дискретное косинусное преобразование

Прямое и обратное дискретное косинусное преобразование реализуется блоками DCT и IDCT. Рисунок 14.44 поясняет технику применения этих блоков. Эти блоки не имеют параметров — их окна установки параметров являются чисто информационными (рис 14.44)

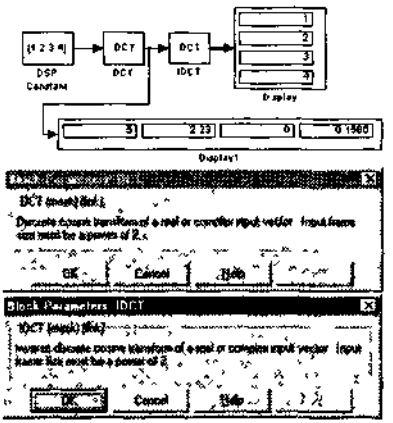


Рис. 14.44. Пример преобразований DCT и IDCT

Блок Analytic Signal

Блок Analytic Signal служит для преобразования каждого канала в общем случае матрицы размером $M \times N$ в комплексный аналитический сигнал вида $y = u + j\,H\{u\}$, где $H\{u\}$ — преобразование Гильберта. Простой пример такого преобразования для вектора [0 1 2 –3] представлен на рис 14.45.

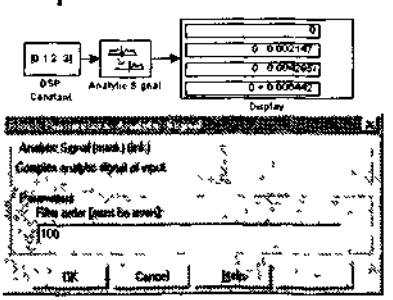


Рис. 14.45. Пример преобразований DCT и IDCT

Окно параметров этого блока также представлено на рис. 14.45. Единственным параметром блока является порядок фильтра Filter order.

Другие блоки раздела Transforms

На рис. 14.46 показано применение других блоков раздела Transform.

Блок |IFFT|^2 вычисляет квадрат модуля данных обратного преобразования Фурье

Блоки Complex Cepstrum и Real Cepstrum служат для осуществления следующей операции:

```
y = real(ifft(log(abs(fft(u Mo)))))
```

или в более компактном виде:

```
y = rceps(u Mo)
```

На выходе создается матрица с вещественными элементами размером $Mo \times N$, где Mo задается по данным FFT-преобразования.

На рис. 14.46 представлены также окна параметров этих блоков.

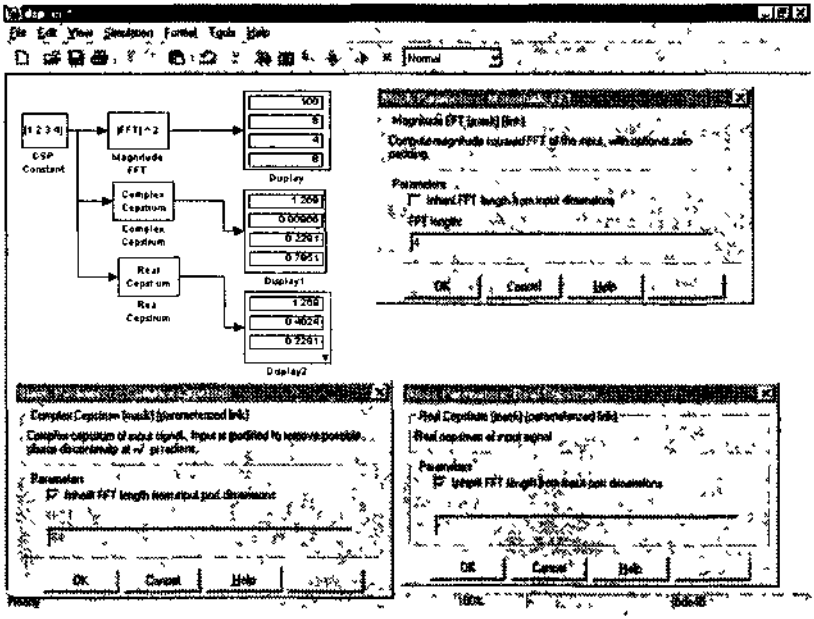


Рис. 14.46. Примеры применения блоков |IFFT|^2, Complex Cepstrum и Real Cepstrum

В каждом блоке всего два параметра: параметр Inherit FFT length from input port dimension делает длину вектора для FFT-преобразования равной длине входного вектора, а FFT length позволяет задать длину вектора произвольно (64 по умолчанию).

Статистическая обработка данных (раздел DSP Statistics)

Обзор раздела DSP Statistics

Окно с блоками раздела DSP Statistics показано на рис. 14.47. В нем представлено 12 блоков, выполняющих типовые статистические вычисления, лежащие в основе статистической обработки данных и сигналов.

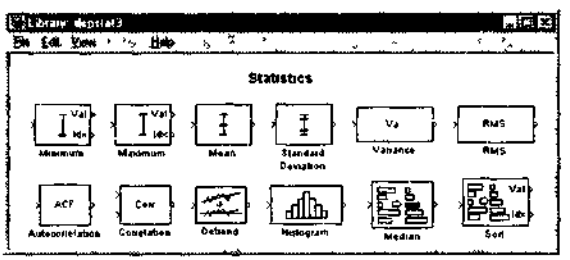


Рис. 14.47. Окно раздела DSP Statistics

В состав этого раздела входят следующие блоки.

- Minimum — выделение элемента с минимальным значением и его индекса;
- Maximum — выделение элемента с максимальным значением и его индекса;
- Mean — вычисление среднего;
- Standard Deviation — вычисление стандартного отклонения;
- Variance — вычисление вариации;
- RMS — среднеквадратическое значение для элементов входного вектора (Root Mean Square);
- Autocorrelation — вычисление автокорреляции;
- Correlation — вычисление кросс-корреляционной функции для столбцов входных данных;

- Detrend — удаление линейной составляющей из вектора;
- Histogramm — подготовка данных для гистограммы;
- Median — вычисление медианы;
- Sort — сортировка и вывод данных с их индексами.

Простейшие статистические вычисления

Ряд блоков раздела DSP Statistics реализуют простейшие статистические вычисления. Примеры их применения представлены на рис 14 48.

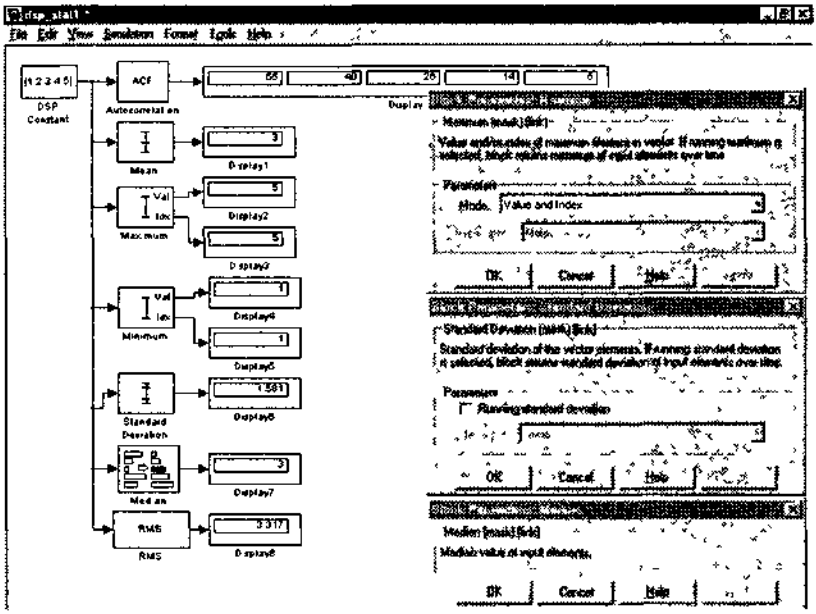


Рис. 14.48. Блоки простейших статистических операций

Ввиду очевидности этих вычислений их подробное описание (как и описание типовых параметров большинства блоков) опущено.

Блоки статистических преобразований

Будем называть блоками статистических преобразований те блоки, которые преобразуют входные данные множественного типа в выходные данные также множественного типа. Работа трех таких бло-

ков (Sort, Detrend и Histogram) представлена на рис. 14.49. Там же показаны окна их параметров.

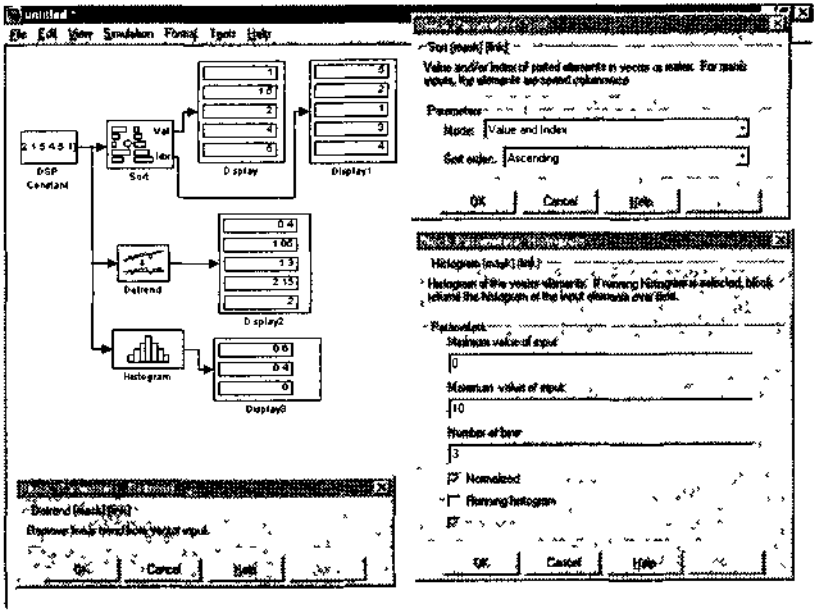


Рис. 14.49. Применение блоков Sort, Detrend и Histogramm

Блок сортировки Sort возвращает отсортированный входной вектор. Порядок сортировки задается параметром Sort Order. Параметр Mode позволяет выводить не только отсортированный вектор, но и индексы элементов этого вектора.

Блок Detrend преобразует входной вектор данных в вектор, в котором отсутствует линейная составляющая. Блок Histogram распределяет данные по заданному числу интервалов и находит количество входящих в них данных (как в абсолютном виде, так и в относительном). Эти данные могут использоваться для построения в дальнейшем графических гистограмм.

Блок кросс-корреляции Correlation

Блок Correlation вычисляет вектор кросс-корреляции для двух векторов входных данных. Его работа иллюстрируется рис. 14.50. Параметров этот блок не имеет.

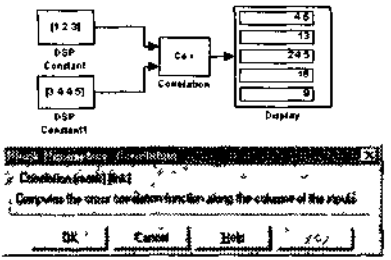


Рис. 14.50. Применение блока Correlation

Фильтрация сигналов (раздел Filtering)

Обзор раздела Filtering

Раздел Filtering (рис. 14.51) библиотеки пакета DSP является одним из самых крупных. Однако, поскольку моделирование фильтров представляет интерес для ограниченного круга читателей, мы рассмотрим этот раздел библиотеки обзорно.

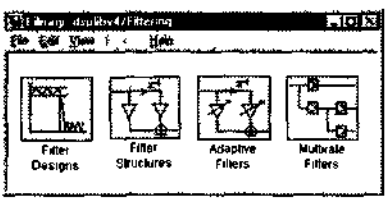


Рис. 14.51. Состав раздела Filtering

Раздел Filtering содержит четыре подраздела:

- Filter Designs — конструирование фильтров разного вида;
- Filter Structures — структуры фильтров разного вида;
- Adaptive Filter — адаптивные фильтры;
- Multirate Filter — многополосные фильтры.

Подраздел Filter Designs

Состав подраздела Filter Designs представлен на рис. 15.52. Он содержит семь блоков фильтров, назначение которых вытекает из их имен.

Блоки позволяют реализовать фильтры типа FIR (с конечной импульсной характеристикой), IIR (с бесконечной импульсной характеристикой) и аналоговые фильтры.

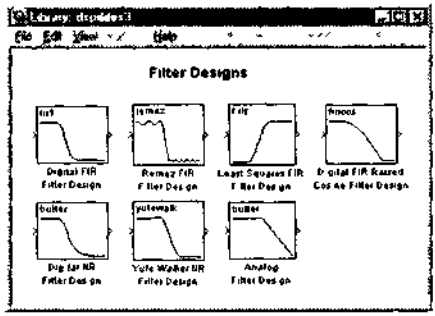


Рис. 14.52. Блоки подраздела Filter Design

Подраздел библиотеки Filter Structures

Подраздел Filter Structures (рис. 14.53) содержит восемь блоков фильтров. Их назначение также очевидно из имен блоков. Блоки этого раздела реализуют основные типы фильтрации сигналов. Работа некоторых из них рассматривается ниже.

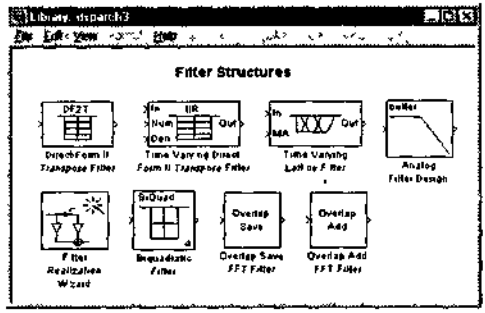


Рис. 14.53. Блоки подраздела Filter Structures

Подраздел Adaptive Filter

Подраздел Adaptive Filter, представленный на рис. 14.54, содержит три блока адаптивных фильтров. Это фильтры Калмана, адаптивные фильтры LMS и RMS.

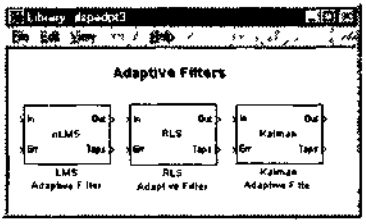


Рис. 14.54. Блоки подраздела Adaptive Filter

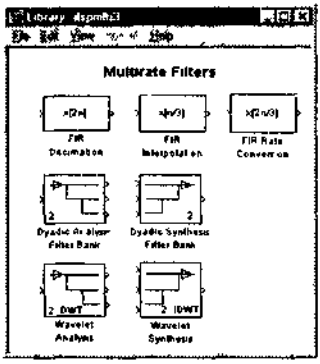


Рис. 14.55. Блоки подраздела Multirate Filter

Подраздел Multirate Filter

Подраздел Multirate Filter (рис. 14.55) содержит семь блоков фильтров специального назначения. Здесь представлены варианты FIR-фильтров (с конечной импульсной характеристикой), фильтры на основе FFT (БПФ) и wavelet-преобразований. Последние являются представителями сравнительно нового вида преобразований, основанного на применении базисных функций, ограниченных (в отличие от гармонических колебаний при Фурье-преобразовании) во времени. Благодаря этому wavelet-преобразования лучше подходят к сигналам, например импульсным или видео, ограниченным во времени и имеющим перепады уровня.

Примеры моделирования фильтров рассматриваются ниже.

Примеры применения пакета DSP

Доступ к демонстрационным примерам пакета DSP

Доступ к демонстрационным примерам пакета DSP возможен двумя способами:

- из окна демонстрационных примеров MATLAB Demo Window, открывающегося командой меню Help ▸ Demos (рис. 14.56);
- из папки toolbox\dspblks\dspdemos

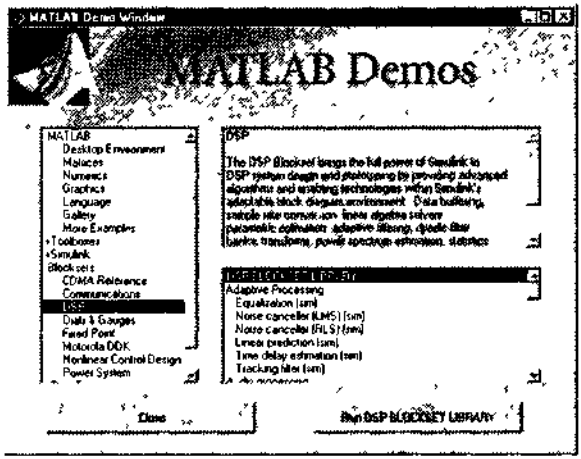


Рис. 14.56. *Окно демонстрационных примеров MATLAB*

В заголовке окон представленных ниже моделей выводится имя файла демонстрационного примера, находящегося в папке toolbox \dspblks\dspdemos.

14 Адаптивная дельта-импульсная кодовая модуляция

Пакет DSP открывает широчайшие возможности в моделировании устройств и систем связи и телекоммуникаций. Ниже рассмотрены некоторые примеры применения пакета DSP. Их основная цель — показать возможности пакета DSP.

В современной технике цифровой связи широкое применение находит так называемая дельта-модуляция (DM). При этом виде модуляции кодируется приращение сигнала. Простейший вид такой модуляции — линейная дельта-модуляция (LDM).

На рис. 14.57 дан пример устройства, которое осуществляет адаптивную дельта-импульсную кодовую модуляцию (ADPCM), заметно превосходящую LDM по качеству передачи быстроизменяющихся сигналов. На рисунке показаны основная модель тракта модуляции и подсистемы адаптивного модулятора и демодулятора, построенные на основе блоков пакета DSP.

На рис. 14.58 показаны результаты моделирования в виде осциллограмм. Верхняя осциллограмма показывает исходный сигнал и наложенный на него сигнал, прошедший тракт модуляции — демодуляции.

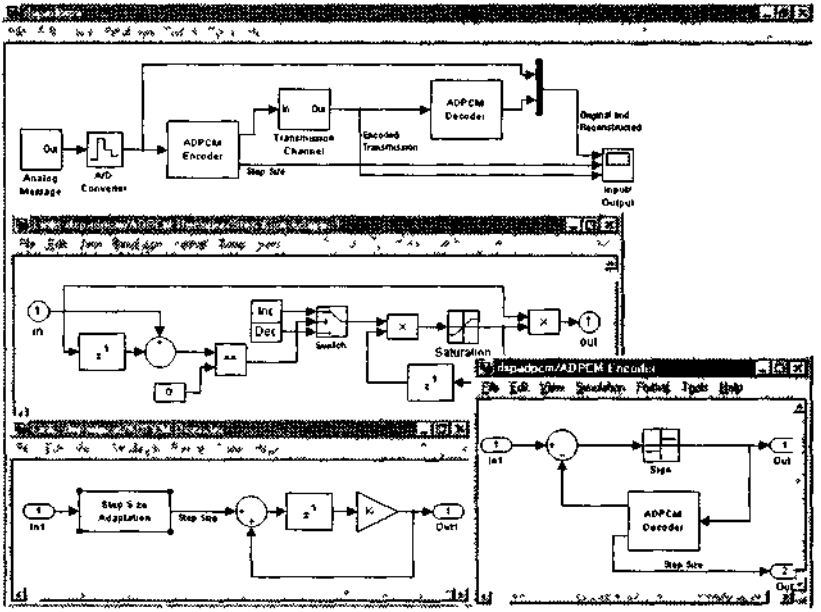


Рис. 14.57. Пример моделирования тракта с ADPCM

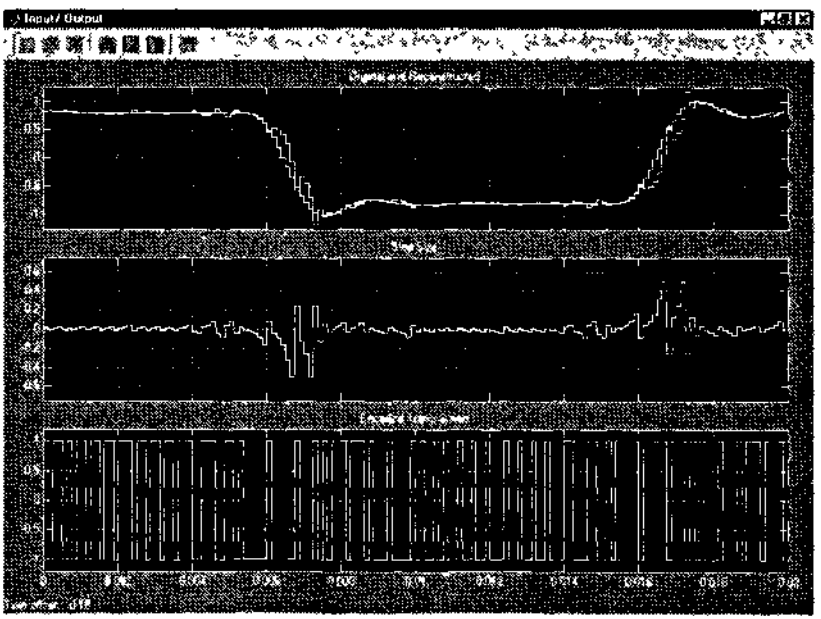


Рис. 14.58. Осциллограммы модели тракта ADPCM

Дельта-модуляция типа CVSD

Модель тракта еще одного вида дельта-модуляции (типа CVSD) представлена на рис. 14.59. Полное название этого вида модуляции представлено на рисунке. Там же приведены результаты моделирования тракта, представленные в виде осциллограмм.

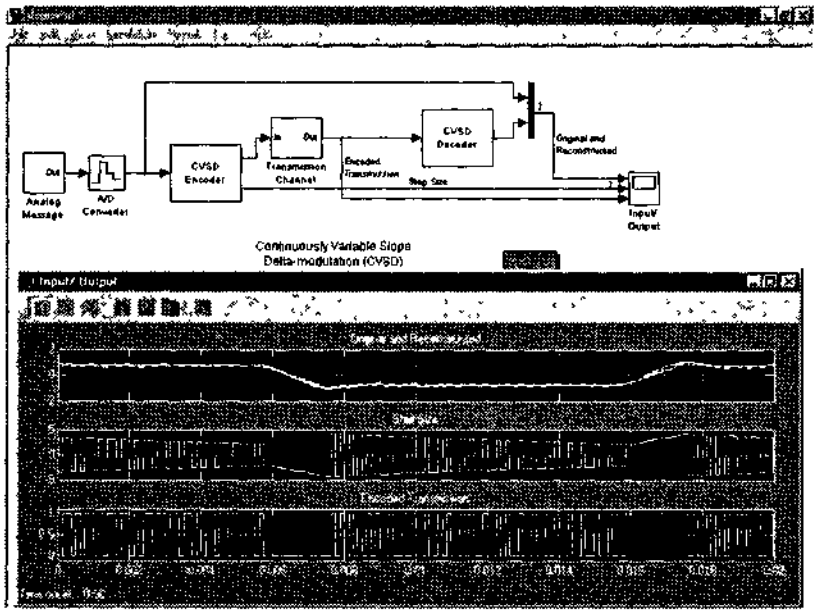


Рис. 14.59. Модель тракта дельта-модуляция типа CVSD

Здесь подсистемы модулятора и демодулятора не раскрыты. Читатель может просмотреть их реализацию самостоятельно, поскольку эти подсистемы являются типичными масками.

Сравнение трех видов дельта-модуляции

Простая модель, иллюстрирующая выполнение дельта-модуляции сразу трех типов (LDM, CVSD и ADPCM), представлена на рис. 14.60. На этот раз в качестве тестового (входного) сигнала использован сигнал свип-генератора в виде «синусоиды» с нарастающей частотой. Этот сигнал имеет довольно широкий спектр и хорошо подходит на роль тестового сигнала, близкого к реальным сигналам.

Осциллограммы модулирующей функции и сигнала на каждом выходе трактов модуляции — демодуляции даны на рис. 14.61.

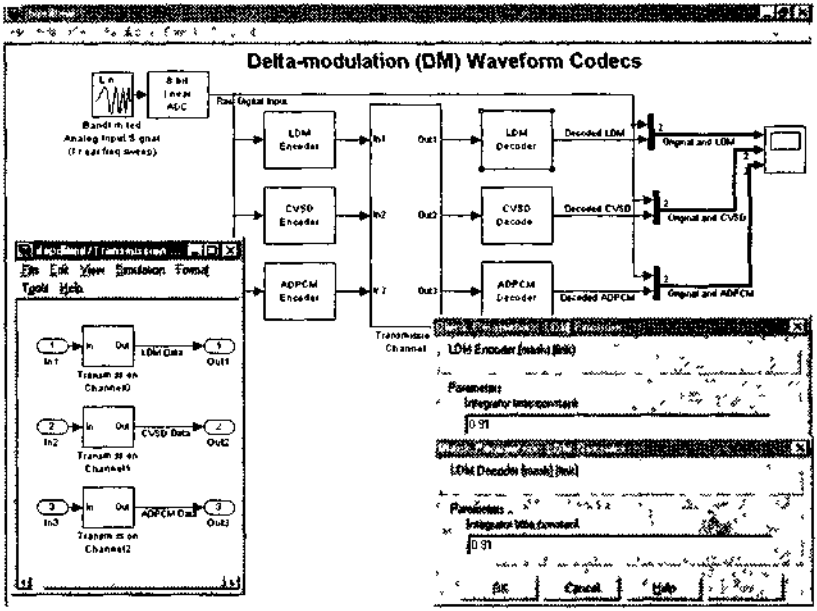


Рис. 14.60. Модель тракта дельта-модуляции трех типов

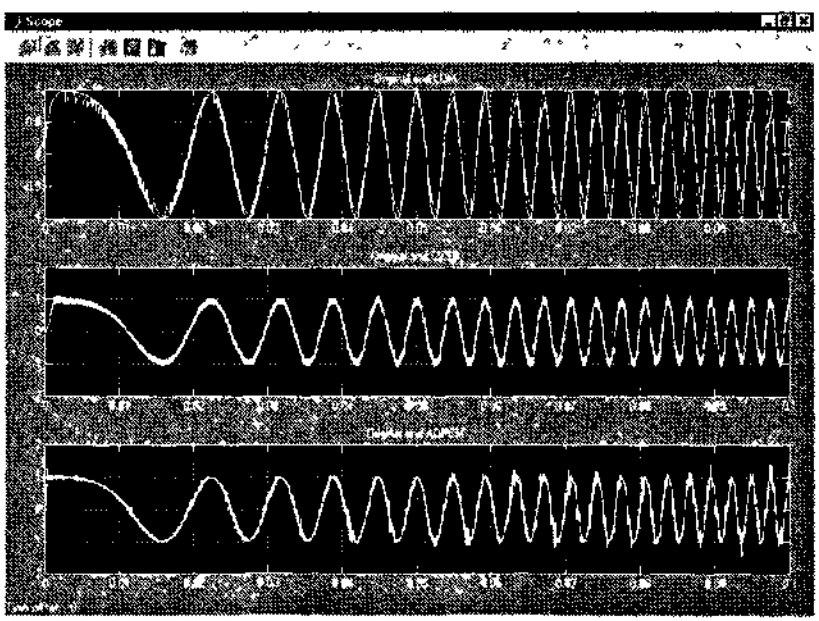


Рис. 14.61. Результаты моделирования устройства рис. 14 60

Результаты моделирования довольно очевидны: LDM-метод дает самую большую погрешность и длительный начальный переходной процесс, CVSD-метод дает намного лучшие результаты, а ADPCM имеет наименьшее время начального переходного процесса, но иногда создает очень короткие выбросы на выходе (то есть требует фильтрации выходного сигнала, в принципе достаточно простой)

Однополосная модуляция (SSB)

Как известно, спектр амплитудно-модулированного колебания состоит из линии несущей частоты и двух боковых полос. Обе они несут по существу одинаковую информацию о модулирующем сигнале. Удаление одной из боковых частот уменьшает вдвое полосу частот, которую должен иметь приемник сигнала При этом снижается уровень помех при приеме. Дополнительный выигрыш в помехозащищенности достигается путем удаления (полного или частичного) несущей частоты, что при заданной мощности передатчика означает существенное повышение уровня сигнала.

К сожалению, модуляция на одной боковой полосе (SSB) с подавлением несущей частоты реализуется технически сложно и моделирование этого процесса с помощью средств пакета DSP и Simulink имеет большое значение. Рисунок 14.62 демонстрирует простейшую модель системы SSB — Simple Basic Version. В ней используются источники синусоидальных сигналов пакета DSP и блоки математических операций, выделяющие боковые полосы (балансный метод SSB).

Результаты моделирования представлены спектрами боковых полос и осциллограммами сигналов этих полос.

Еще одна система SSB представлена на рис. 14.63 — Frame Basic Version. Здесь наряду с математическими операциями широко используются блоки матричных операций. Результаты моделирования наглядно представлены спектрами полос и временными зависимостями сигналов.

Следует отметить, что в этих примерах реализована возможность наблюдения за процессом однополосной модуляции в динамике. Представленные результаты получены как стоп-кадры анимационной картины модуляции при ограниченном времени моделирования.

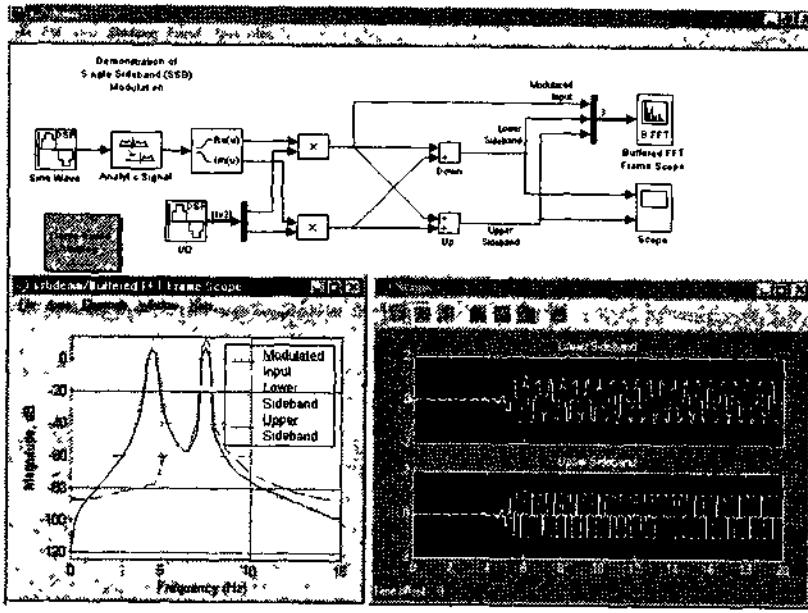


Рис. 14.62. Модель однополосной модуляции (SSB) — Simple Basic Version

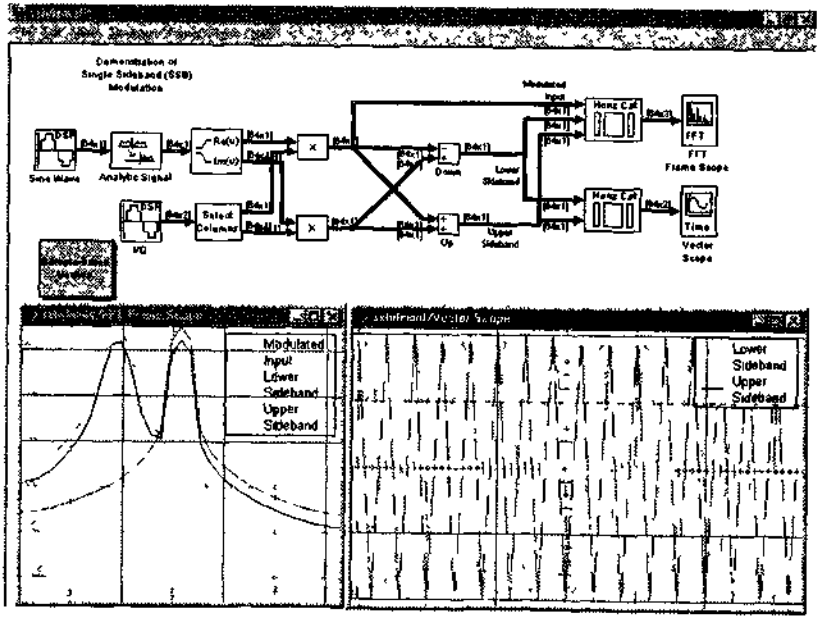


Рис. 14.63. Модель однополосной модуляции (SSB) — Frame Basic Version

FIR-интерполяция синусоидального сигнала

В основе техники цифрового преобразования и фильтрации сигналов лежит возможность квантования аналоговых сигналов и дальнейшего восстановления их исходной формы. Для восстановления формы сигнала широко используется техника интерполяции с ограничением скорости выходного сигнала. Она реализуется с помощью фильтрующих цепей с конечной импульсной характеристикой (FIR или КИХ) различного порядка.

Рисунок 14.64 иллюстрирует технику задержки квантованного синусоидального сигнала и восстановления его на основе FIR-интерполяции. На этом рисунке показаны также окна параметров блоков интерполяции.

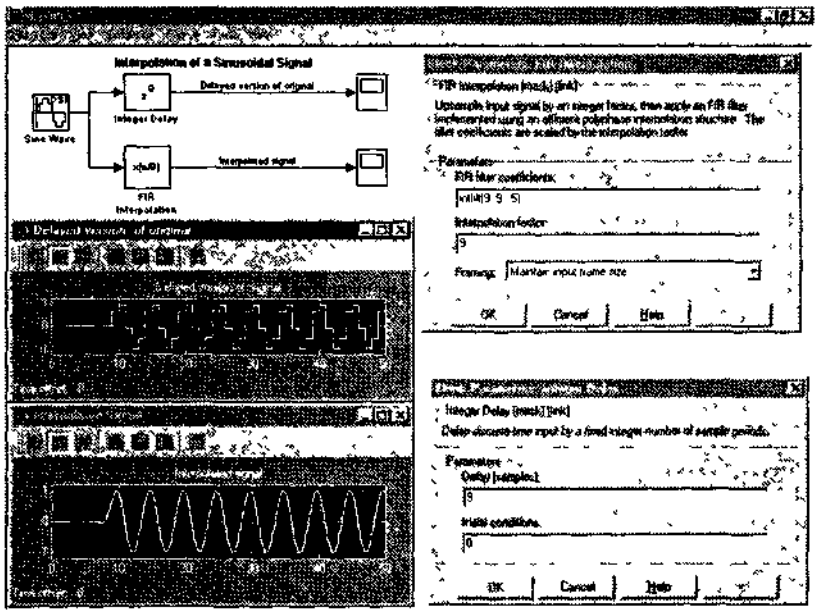


Рис. 14.64. Задержка и FIR-интерполяции квантованного синусоидального сигнала

Нетрудно заметить, что FIR-интерполятор превосходно восстанавливает синусоидальный сигнал, что свидетельствует о высокой точности представления этого сигнала в цифровом виде. При этом харак-

терно, что интерполированный сигнал появляется со значительной задержкой. Именно поэтому для удобства сравнения в тракт контроля квантованного сигнала искусственно введена задержка.

На рис. 16.64 представлена простая версия этого примера (Simple-base Version). Активизируя пиктограмму Frame-base Version, можно просмотреть еще один пример интерполяции. Он отличается от описанного только введением в тракт задержки и интерполяции блоков Unbuffer.

Моделирование адаптивного фильтра

Цифровые средства открывают путь к широкому применению адаптивных фильтров, свойства которых подстраиваются под особенности входного сигнала — особенно если они априорно известны. Рисунок 16.65 иллюстрирует технику адаптивной фильтрации для смеси синусоидального сигнала с шумом.

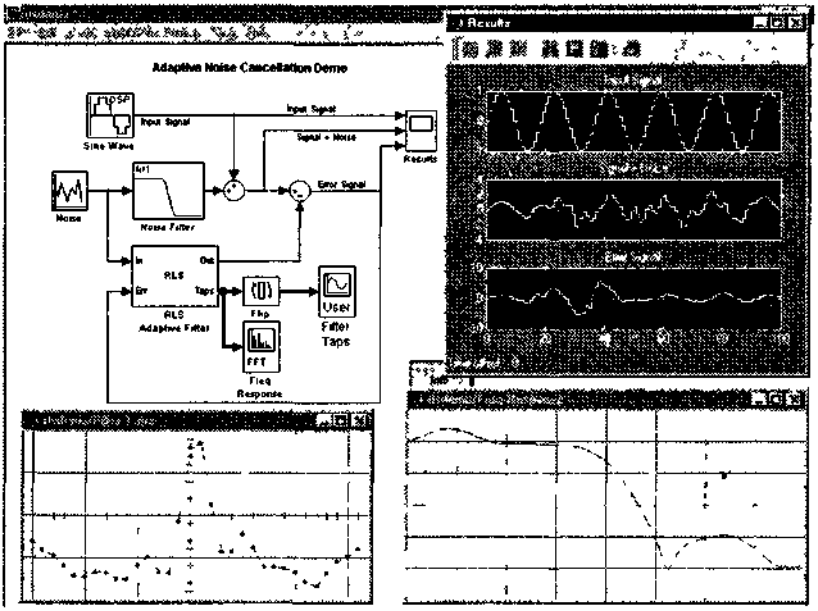


Рис. 14.65. Моделирование адаптивного RLS-фильтра

Представленные на рис. 14.65 характеристики фильтра даны в виде стоп-кадра. На самом деле можно видеть, как они меняются во време-

ни, подстраиваясь под фильтрацию сигнала. Осциллограммы дают представление о временных зависимостях сигнала и о погрешности, вызванной шумами.

Моделирование многополосных фильтров

Файл dspmrf_menu открывает окно с пиктограммами ряда простых демонстрационных примеров на построение многополосных фильтров. Это окно представлено на рис. 14.66. Фильтры заданы в двух вариантах — с фреймами и без фреймов.

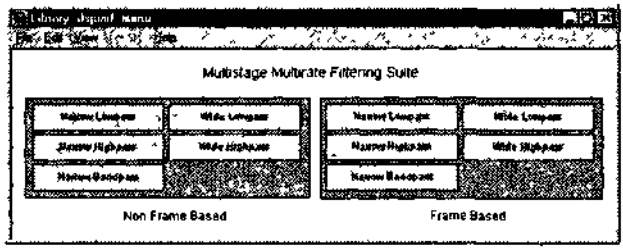


Рис. 14.66. Блоки моделирования многополосных фильтров

Рисунок 14.67 показывает пример моделирования узкополосного полосового фильтра.

Моделирование аудиосистем

Имеется ряд интересных примеров на моделирование аудиосистем. На рис 14.68 показан пример моделирования системы спектрального сжатия звуковых сигналов (сигнала рабочего пространства и синусоиды). Степень сжатия может устанавливаться ползунком. Входной и выходной сигналы, также представленные на рисунке, практически идентичны, что свидетельствует о малой потере качества воспроизведения звука.

Другой пример (Audio Flanger — рис. 14.69) иллюстрирует технику спектрального анализа сложного сигнала с построением спектрограмм в виде распределения яркости графического представления сигнала на плоскости. Такое представление нередко позволяет обнаруживать тонкости спектрального сигнала, исчезающие при обычном представлении спектрограмм. Это хорошо видно из сравнения спектрограмм входного и выходного сигналов.

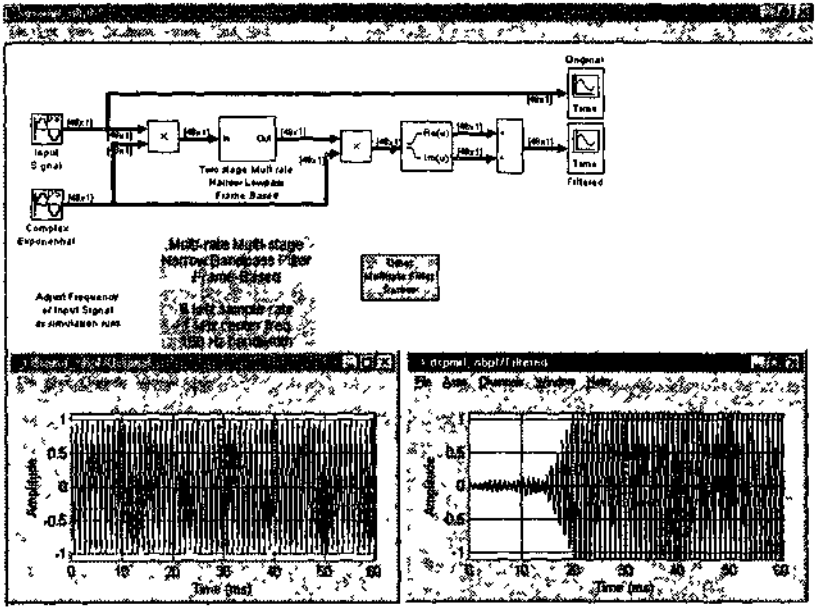


Рис. 14.67. Пример моделирования узкополосного фильтра

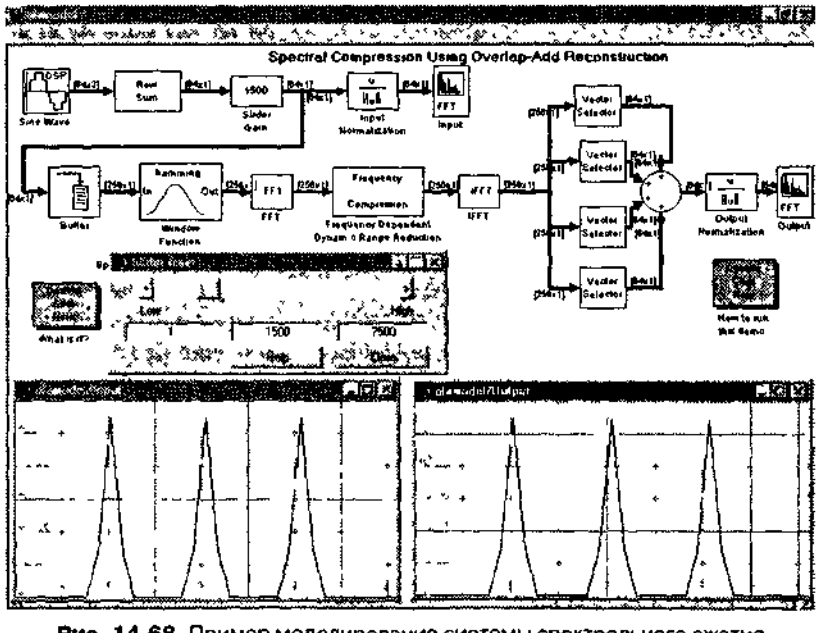


Рис. 14.68. Пример моделирования системы спектрального сжатия звукового сигнала

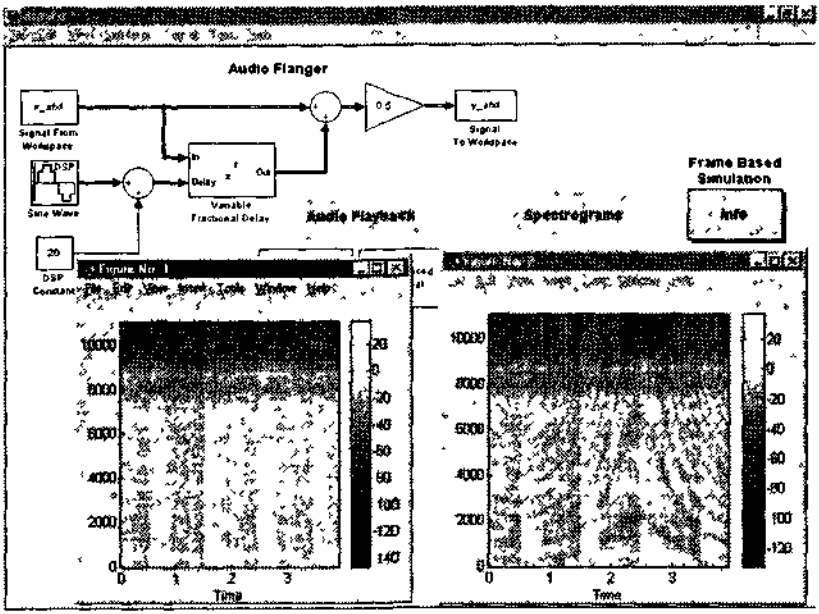


Рис. 14.69. Пример построения спектрограмм в виде распределения яркости образа сигналов на плоскости

Активизируя пиктограмму PC Windows Demo (она на рис 14.69 скрыта правой спектрограммой), можно открыть окно с версией этого примера, которая позволяет обрабатывать сигнал микрофона, если ваш ПК оснащен микрофоном и звуковой картой.

Быстрый спектральный анализ

Рисунок 14.70 показывает модель, основанную на быстром спектральном анализе, базирующемся на технике БПФ (FFT). Она реализуется блоком Short-Time FFT.

Здесь результат моделирования показан как в виде обычной спектрограммы, так и с использованием ассоциации между спектральной плотностью и цветом.

Нетрудно заметить, что хотя обычная спектрограмма более привычна, спектрограмма с использованием ассоциации между спектральной плотностью сигнала и цветом более информативна (к сожалению, черно-белый рисунок в книге начисто уничтожает это различие).

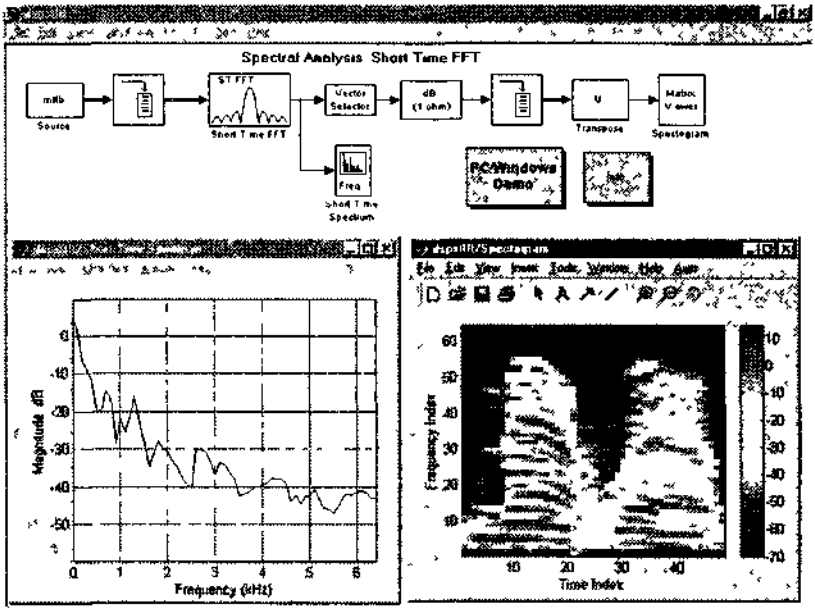


Рис. 14.70. Пример быстрого спектрального анализа

Использование техники wavelet-преобразований

Как уже отмечалось, в последнее время широкое применение находит техника wavelet-преобразований. В частности, она уже применяется в популярном стандарте сжатия видеоизображений MPEG-4, используемом для записи видеофильмов на CD-ROM с минимальной потерей качества изображения при больших степенях сжатия. Минимально необходимый набор средств для wavelet-преобразований входит в пакет расширения DSP, а достаточно представительный wavelet-инструментарий имеется в пакете расширения Wavelet (его описание не входит в задачи этой книги).

Wavelet-преобразование имеет много общего с преобразованием Фурье. Так, они имеют похожие экспоненциальные представления и технику быстрого wavelet-преобразования, похожую по своей реализации на технику быстрого преобразования Фурье (БПФ). Но между ними есть и принципиальное различие. В рядах Фурье используются гармоники с заданными номерами, кратными частоте

сигнала (или частоте первой гармоники). Каждая гармоника является синусоидой, то есть сигналом, определенным на всей оси времени. Это и обусловливает недостатки рядов Фурье при представлении импульсных сигналов, определенных в ограниченном интервале времени и содержащих разрывы.

Wavelet-представление оперирует базовыми функциями типа «выпуклости», которые характеризуются сдвигом во времени, положением и сдвигом положения. Вместо номеров гармоник вводятся wavelet-коэффициенты, на которые умножаются базовые функции.

Интерполяция с применением wavelet-преобразований

К сожалению, вопросы теоретического описания этих преобразований настолько сложны, что им не место в этом демонстрационном разделе. А потому ограничимся практическим примером синтеза сигнала, полученного из рабочего пространства, с применением техники wavelet-интерполяции. Этот простой пример представлен на рис. 14.71. Под основной моделью дана подсистема задания wavelet-коэффициентов.

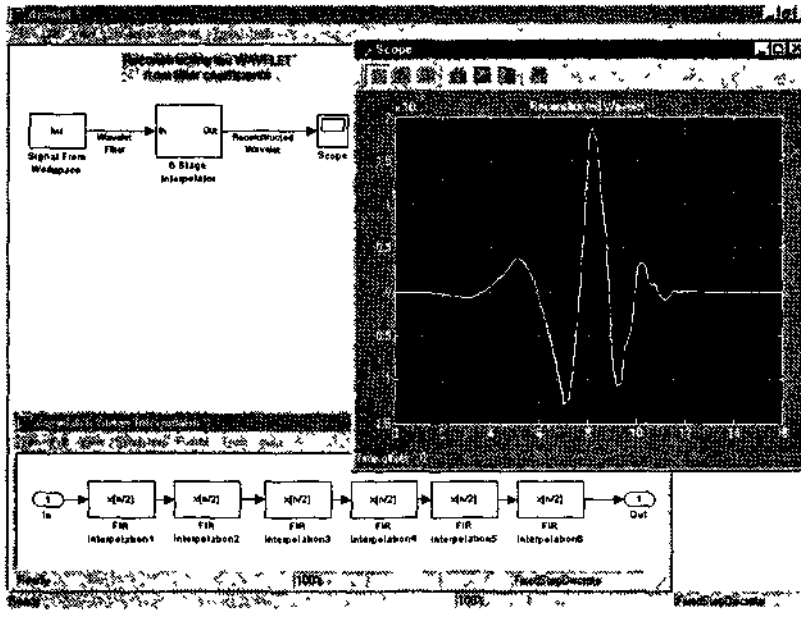


Рис. 14.71. Пример восстановления сигнала на основе wavelet-интерполяции

Реконструкция сигнала на основе wavelet-преобразования

Более сложный пример реконструкции сигнала на основе wavelet-преобразования представлен на рис. 14.72. Здесь преобразованиям подвергается сигнал в виде зашумленной «синусоиды» с нарастающей амплитудой и периодом. Такой сигнал неплохо представляет реальные сигналы в системах связи.

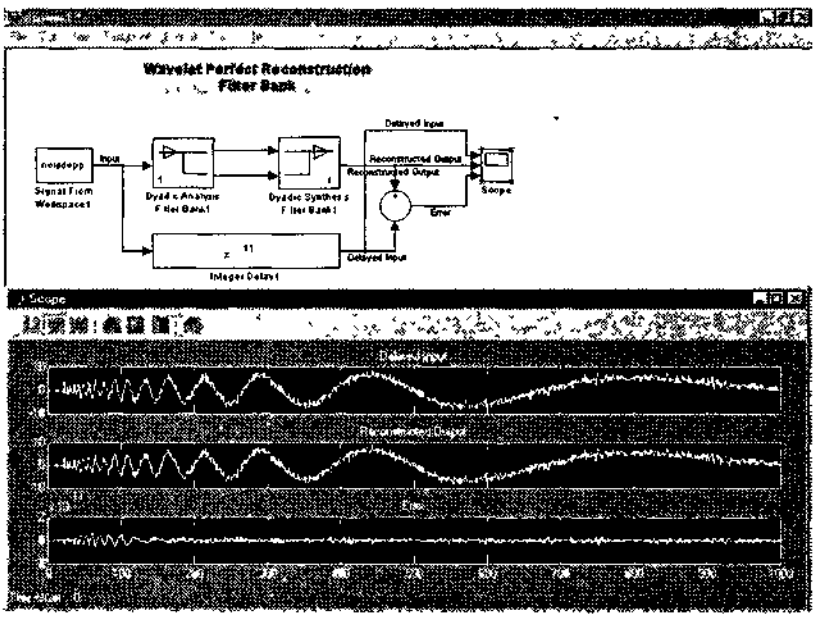


Рис. 14.72. Пример wavelet-преобразований для сложного сигнала

Осциллограммы сигналов на входе и выходе, а также погрешность восстановления также представлены на рис. 14.72.

Моделирование трехканального wavelet-мультиплексора

Модель трехканального wavelet-мультиплексора (рис. 14.73) наглядно демонстрирует технику wavelet-преобразований сразу для трех видов сигналов.

Нетрудно заметить, что во всех случаях достигается великолепное восстановление исходной формы сигнала (см. соответствующие ос-

циллограммы) При этом погрешность восстановления ничтожно мала Особенно впечатляет прекрасное восстановление формы прямоугольных импульсов — случаи, где Фурье-преобразование дает плохие результаты из-за возникновения эффекта Гиббса в моменты скачкообразного изменения сигнала

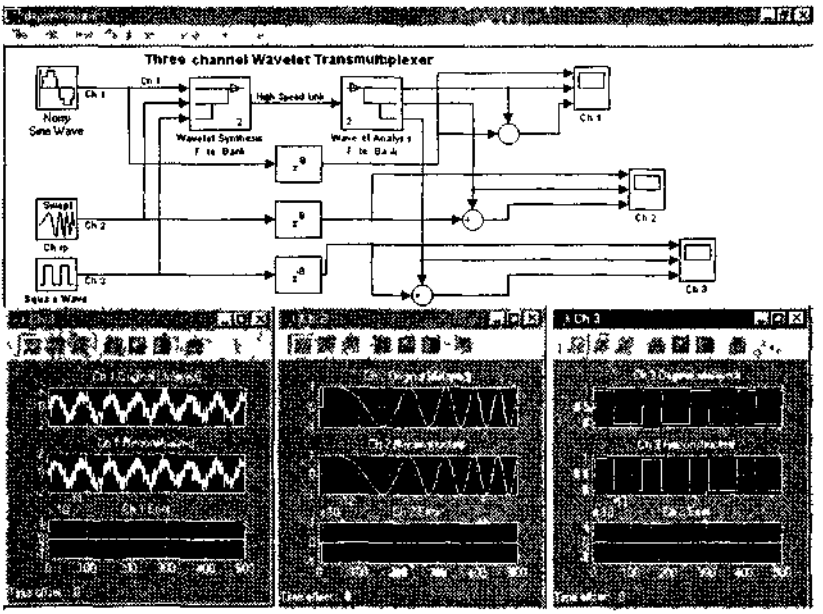


Рис. 14.73. Модель wavelet-мультиплексора

Очистка сигнала от шумов на основе wavelet-преобразований

В заключение приведем еще один пример эффективного применения wavelet-преобразований — для очистки сигнала от шума Модель системы, выполняющей эту функцию, представлена на рис 14 74

С применяемыми в этой модели блоками и работой этой модели заинтересованный читатель может разобраться самостоятельно

Моделирование приемника сигналов точного времени

Рассмотрим демонстрационный пример на моделирование приемника сигналов точного времени (WWW-приемника) (Сходство

названия «WWW» с соответствующей службой в Интернете чисто случайное)

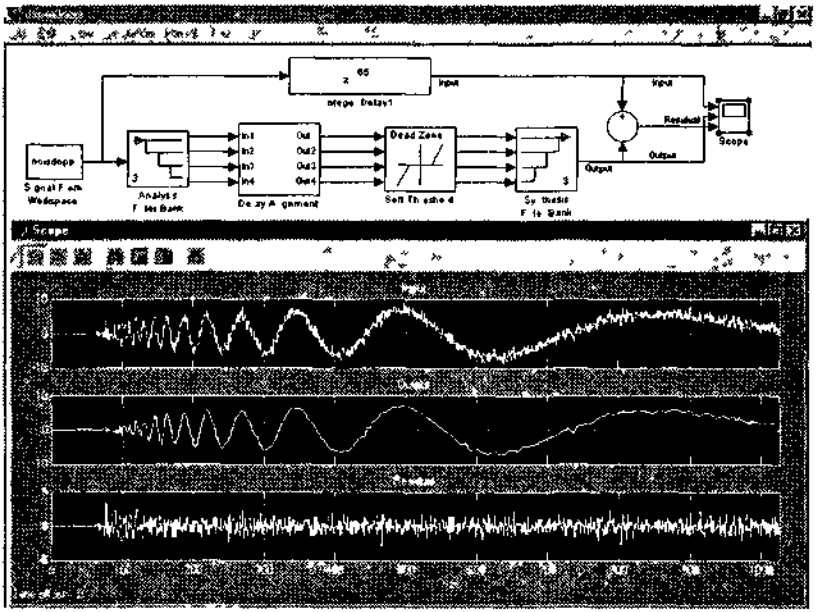


Рис. 14.74. *Пример применения wavelet-преобразований для очистки* сложного сигнала от шума

Модель WWW-приемника

Ряд радиостанций передают сигналы точного времени на частотах 5, 10, 15 МГц и т д Эти сигналы передаются в виде звуковых кодов Задача WWW-приемника заключается в том, чтобы принять сигналы эталонного времени, выделить коды и по ним с максимальной точностью определить момент времени, например 00 часов, 00 минут, 00 секунд Модель WWW-приемника представлена на рис 14 75

Собственно говоря, модель реализует не весь приемник (его высокочастотная часть вообще не представлена), а только тракт обработки акустических сигналов (кодов) времени Обратите внимание на оформление интерфейса модели — вывод окна с датой и текущим временем, а также диаграмму принимаемого сигнала, представляющую собой развертку во времени его кодов В левом нижнем углу представлена подсистема декодера приемника

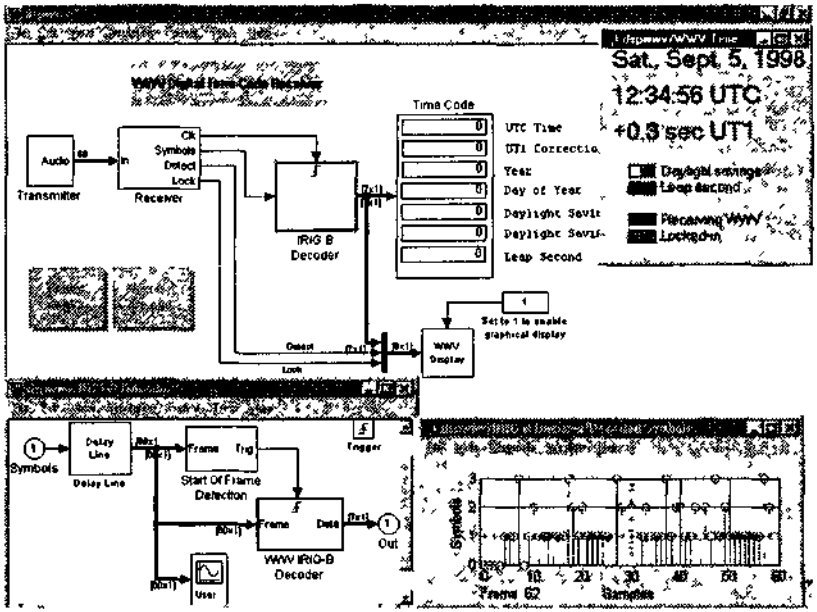


Рис. 14.75. Модель приемника сигналов точного времени

14.4 Подсистема приемника акустических кодов

Наиболее оригинальное решение в данной модели — это приемник акустических кодов, представляющий собой подсистему, логика управления которой задается SF-диаграммой. Для этого в подсистему включен SF-блок Symbol Sync (рис. 14.76).

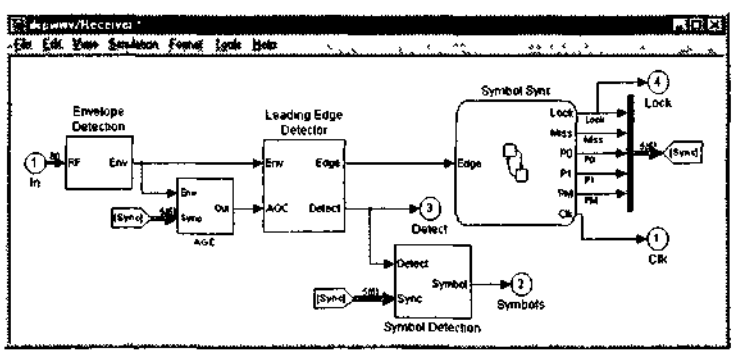


Рис. 14.76. Подсистема приемника акустических кодов

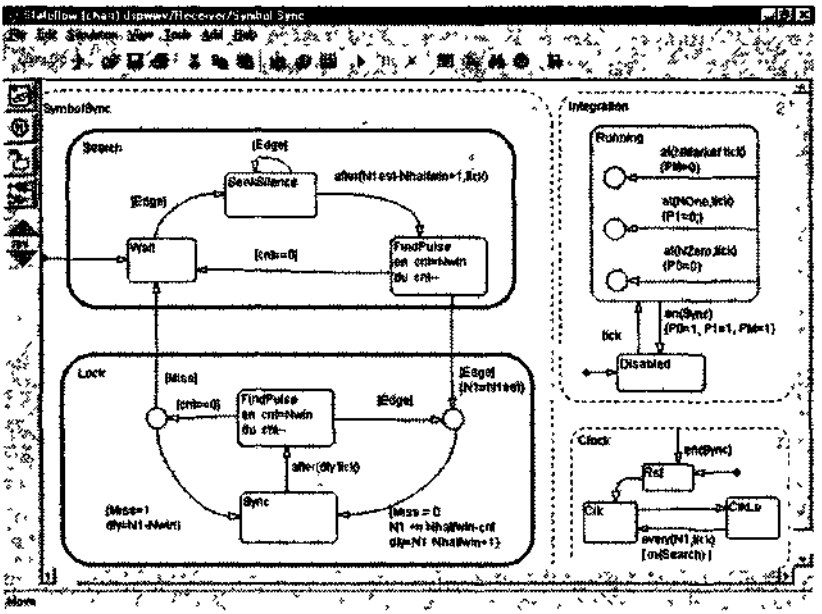


Рис. 14.77. SF-диаграмма приемника сигналов точного времени

Данный пример наглядно иллюстрирует одновременную работу системы MATLAB + Simulink с пакетами DSP и Stateflow.

SF-диаграмма приемника, задающая логику его работы, представлена на рис. 14.77.

Глава 15. Пакет Power System Blockset

- Назначение пакета
- Параметры и единицы их измерения
- Библиотека источников электрической энергии Electrical Sources
- Соединительные элементы
- Библиотека компонентов
- Коммутирующие элементы энергетической электроники
- Моделирование электрических машин и схем управления ими
- Дополнительные возможности пакета Power System

Назначение и состав пакета

Назначение пакета Power System Blockset 2.0

Новая, существенно переработанная и дополненная версия пакета Power System Blockset 2.0 служит для моделирования энергетических (силовых) систем и устройств. Этот пакет занимает особое положение, поскольку больше, чем другие пакеты, ориентирован на моделирование технических устройств и систем вполне конкретного назначения.

Функционально полный набор библиотек и компонентов, превосходный графический интерфейс пользователя системы Simulink, обширные вычислительные возможности базовой матричной системы MATLAB, высокая степень достоверности имитации (моделирования) энергетических устройств и систем и превосходная степень визуализации результатов моделирования — все это сделало пакет Power System Blockset одним из лучших среди пакетов такого рода, доступных для установки на персональных компьютерах с операционной системой Windows 95/98/2000/NT.

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 (рис 15.1) В окне браузера библиотек Simulink можно выбрать библиотеку Power System Blockset.

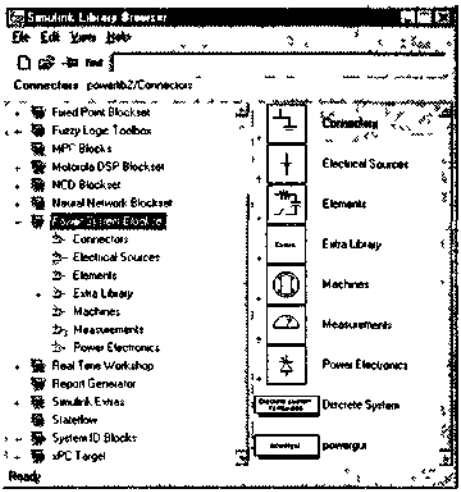


Рис. 15.1. Доступ к библиотекам пакета Power System Blockset

Можно также вызвать окно с разделами библиотеки, выполнив команду `powerlib` в командной строке MATLAB.

Состав библиотек Power System Blockset

В состав Power System Blockset входят библиотеки следующего назначения

- Electrical Sources — источники электрической энергии и сигналов;
- Elements — линейные и нелинейные компоненты электротехнических 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,
- Powergui — графический интерфейс пользователя пакета моделирования энергетических систем

Их применение позволяет создавать модели самых разных энергетических устройств и выполнять их моделирование в режиме работы виртуальных устройств. Это дает наглядное представление о работе реальных устройств.

Параметры и единицы их измерения

При указании параметров компонентов и единиц измерения используются следующие обозначения (в скобках даны обозначения, принятые в русскоязычной литературе):

Параметр	Обозначение	Единицы
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)
Frequence (частота)	Hertz	Hz (Гц)
Active power (активная мощность)	watt	W (Вт)
Apparent power (полная мощность)	volt ampere	VA (ВА)
Reactive power (реактивная мощность)	var	var ()
Impedance (импеданс)	ohm	Ω (Ом)
Resistance (импеданс)	ohm	Ω (Ом)
Inductance (индуктивность)	henry	H (Гн)
Capacity (емкость)	farad	F (Ф)
Flux linkage (поток сцепления)	volt second	V S (В с)
Rotation speed (угловая скорость)	radians per second revolutions per minute	rad/s (рад/с) rpm (пер/мин)
Torque (вращающий момент)	newton meter	N m (Н м)
Inertia (момент инерции)	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 измерения. Некоторые из параметров будут рассмотрены по мере описания моделей.

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.

Пакет Power System Blockset имеет модели источников, позволяющих имитировать реальные источники электроэнергии. Дважды щелкнув мышью на пиктограмме библиотеки Electrical Sources, можно открыть окно этой библиотеки (рис. 15.2).

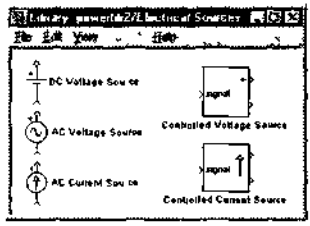


Рис. 15.2. Окно библиотеки источников Electrical Sources

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 источники образуют функционально полный набор источников электрической энергии и минимальный набор источников сигналов (в других расширениях MATLAB кроме источников синусо-

идальных сигналов можно задавать великое множество источников сигналов самой разной формы). Этих наборов вполне достаточно для проектирования энергетических устройств

Источник переменного тока

Источник идеального (с бесконечно большим внутренним сопротивлением) переменного тока с заданной амплитудой, частотой и фазой AC Current Source задает ток, который меняется по синусоидальному закону и описывается выражением:

I=Amplitude × sin(2*pi* × *Frequency* × *t* + *Phase* × *pi*/180)

Рисунок 15.3 показывает пример применения модели источника переменного тока. Здесь показан случай суммирования двух токов в резисторе 50 Ом.

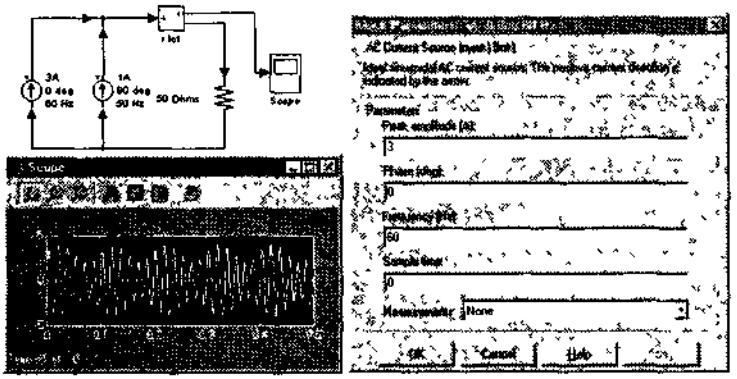


Рис. 15.3. Схема, демонстрирующая суммирование токов от двух источников (слева), и осциллограмма тока в нагрузочном резисторе (справа)

В результате суммирования двух токов с разными частотами (50 и 60 Гц) наблюдается биение результирующего тока с разностной частотой.

Следует отметить, что хотя осциллограмма тока с биениями напоминает модулированный ток, он таковым не является — модуляция возможна в нелинейной системе, а данная простейшая система является линейной.

Этот компонент имеет следующие параметры (рис. 15.3): амплитуда тока (Peak amplitude) в амперах, частота в герцах (Frequency) и фаза (Phase) в градусах (а не в радианах).

Источник напряжения переменного тока

Идеальный источник (с нулевым внутренним сопротивлением) напряжения переменного тока с заданной амплитудой, частотой и фазой AC Voltage Source задает напряжение, меняющееся по синусоидальному закону и описываемое выражением:

$U=Amplitude \times \sin(2\pi \times Frequency \times t + Phase \times \pi/180)$

Параметры источника задаются в окне, представленном на рис 15.4 Амплитуда напряжения (Peak amplitude) задается в вольтах. Приведенный на рис. 15.4 пример демонстрирует суммирование двух напряжений переменного тока на резисторе с сопротивлением 100 Ом.

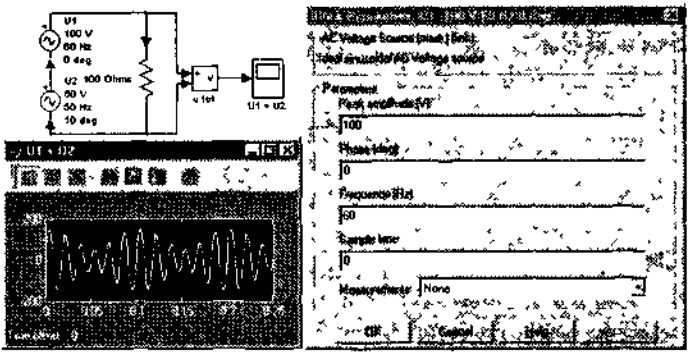


Рис. 15.4. Схема, демонстрирующая суммирование напряжений от двух источников напряжения (слева) и осциллограмма результирующего сигнала

В результате сложения двух переменных напряжений с частотами 50 и 60 Гц видны характерные биения с разностной частотой 10 Гц. Данная цепь также является линейной.

Источник напряжения постоянного тока

Источник напряжения постоянного тока DC Voltage Source задается только одним параметром — выходным напряжением E (такой источник также называется идеальным источником напряжения постоянного тока) (рис. 15.5).

На рис. 15.5 показан также пример применения источника напряжения постоянного тока. В данном случае анализируется заряд конденсатора емкостью 50 мкФ через резистор 50 Ом от источника постоянного напряжения 100 В после включения ключа Breaker спу-

стя время, равное 20 мс после начала моделирования. Осциллограф иллюстрирует экспоненциальный процесс зарядки конденсатора.

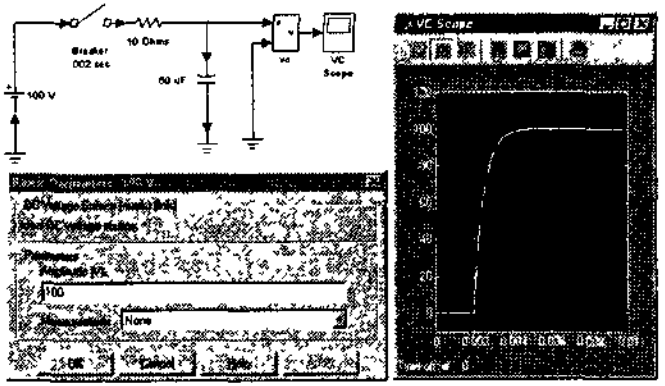


Рис. 15.5. Пример моделирования зарядки конденсатора через резистор от источника постоянного напряжения

В этой модели, демонстрирующей 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 моделей компонентов 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. 15.6) задается только начальный ток (по умолчанию равный 0). При этом временная зависимость выходного тока определяется временной зависимостью управляющего тока. Пример применения управляемого источника тока также представлен на рис. 15.6. Обратите внимание на то, что управляющий сигнал создается операциями суммирования и умножения сигналов двух источников переменного тока и одного источника постоянного тока. Для этого используются блоки системы Simulink, например из математической библиотеки.

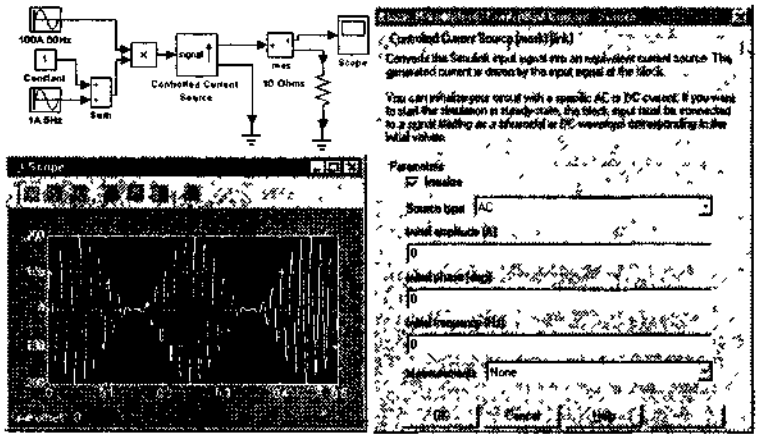


Рис. 15.6. Пример применения управляемого источника тока

Суть работы схемы этого примера вполне очевидна: на резисторе 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 задается только начальное напряжение. Пример применения управляемого источника тока представлен на рис. 15.7.

В приведенном примере формируется сложный управляющий сигнал, представляющий сумму синусоидального напряжения с частотой 60 Гц и амплитудой 120 В с коммутируемым сигналом с частотой 180 Гц и амплитудой 100 В. Этот сигнал используется для управления источником управляемого напряжения, выход которого подключен к RC-цепи. Ее выходной сигнал и подается осциллографом.

Управляемые источники напряжения и тока позволяют моделировать такой важный класс электрических цепей, как параметрические цепи (цепи, параметры которых зависят от времени).

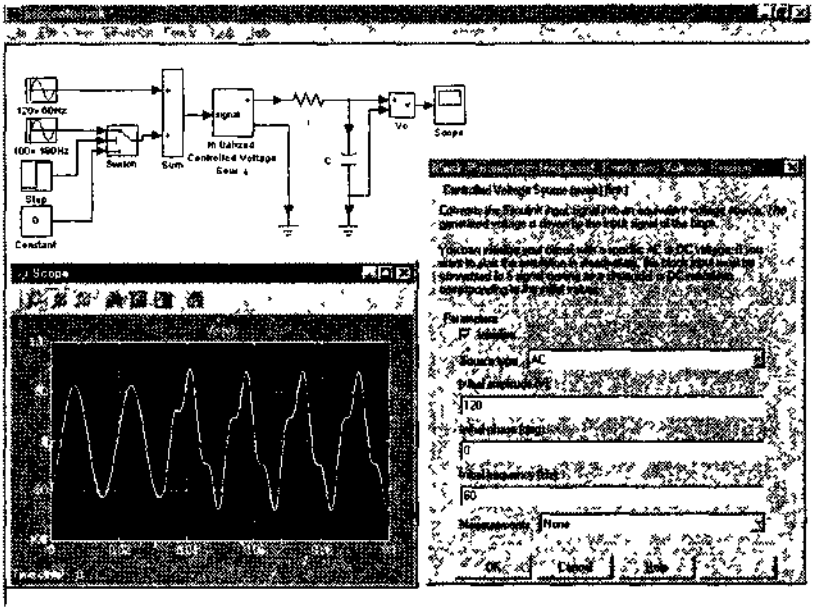


Рис. 15.7. Пример применения управляемого источника напряжения

Соединительные элементы

Состав соединительных элементов

Состав соединительных элементов представлен на рис. 15.8.

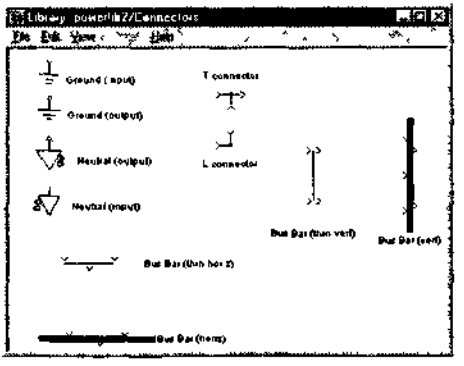


Рис. 15.8. Библиотека соединительных элементов

Большинство соединительных элементов имеют вполне очевидное назначение. Например, заземляющие элементы служат для подключения других элементов к общему проводу — земле (блок Ground). Эти элементы не имеют параметров. Более детально эти элементы рассматривать не имеет смысла ввиду очевидности их назначения.

Нейтраль

Нейтраль (блок Ground) — это общий для всех 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). Он представляется на схеме как большой треугольник с вершиной, обращенной вниз, и номером узла рядом.

Рисунок 15.9 показывает применение заземляющего элемента и нейтралей в схеме трехфазной сети переменного тока

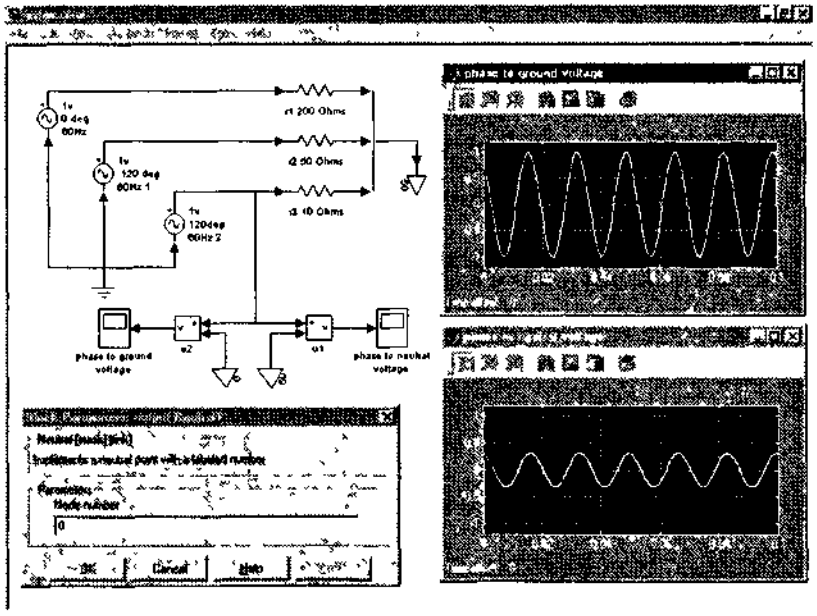


Рис. 15.9. Применение заземляющего элемента и нейтралей

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 электрических цепей компонентами

Блок шин

Блок шин Bus является идеальным переходником от одиночной шины к ряду шин. Окно параметров этого блока (рис. 15.10) имеет всего два параметра: число входных шин и число выходных шин. Пример применения блока шин также дан на рис. 15.10. В этом примере блок шин используется для объединения токов от трех входных цепей в выходной шине. Эти цепи образованы последовательным контуром R1 L1 C1, резистором R2 и конденсатором C2.

В данном случае на модели опущено указание номиналов компонентов. Напоминаем, что они заданы в окнах установки параметров блоков

Библиотека компонентов

Основная библиотека компонентов (рис. 15.11) содержит ряд моделей, имеющих достаточно универсальный характер. С помощью одной модели можно, как правило, создать модели нескольких компонентов.

Эта библиотека содержит несколько характерных компонентов:

- Series RLC Branch — последовательная RLC-цепь;
- Series RLC Load — последовательная RLC-цепь с нагрузкой;
- Parallel RLC Branch — параллельная RLC-цепь;
- Parallel RLC Load — параллельная RLC-цепь с нагрузкой;
- Linear Transformer — линейный трансформатор;
- Saturable Transformer — нелинейный трансформатор;
- Mutual Inductance — блок взаимной индуктивности;
- Surge Arrester — ограничитель пиковых напряжений;
- Breaker — выключатель управляемый;
- PI Section Line — линия с сосредоточенными параметрами;
- Distributed Parameters Line — линия с распределенными параметрами.

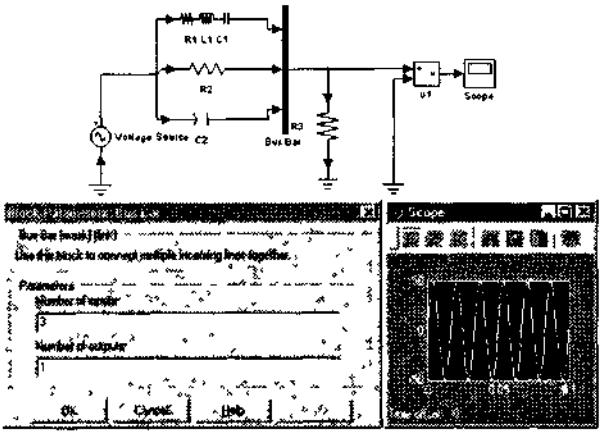


Рис. 15.10. Пример применения блока шин

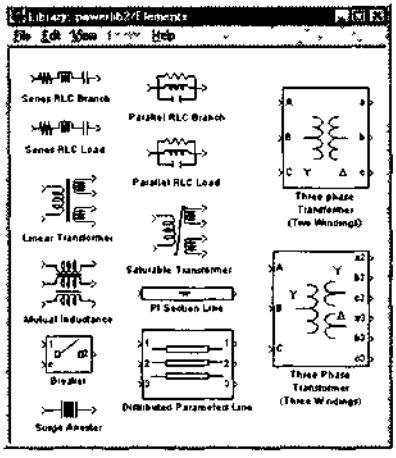


Рис. 15.11. Окно библиотеки компонентов Elements

Последовательные и параллельные LC-цепи

В состав библиотеки входят две последовательные и две параллельные RLC-цепи. Эти цепи (последовательная Series RLC Branch и параллельная Parallel RLC Branch) задаются тремя параметрами: сопротивлением R, индуктивностью L и емкостью C. У так называемых нагрузочных цепей (последовательной Series RLC Load и параллель-

ной Parallel RLC Load) дополнительно задаются допустимые мощности рассеяния активная для резистора и реактивные для индуктивности и конденсатора Ввиду очевидности задания этих параметров ограничимся приведением окна параметров для нагрузочной последовательной RLC-цепи — рис 15.12.

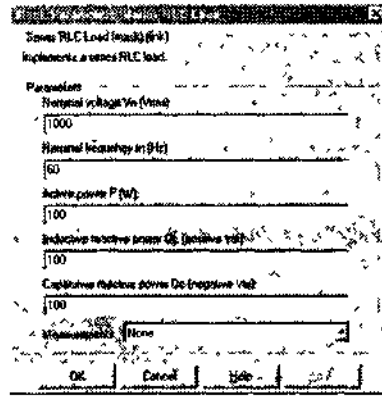


Рис. 15.12. Окно параметров нагрузочной последовательной RLC-цепи

Последовательные и параллельные RLC-цепи могут использоваться для моделирования колебательных контуров и создания эквивалентов нагрузки.

15. Отдельные элементы R, L и C

Для ввода отдельных элементов (резистора R, конденсатора C и индуктивности L) можно использовать любую из RLC-цепей, задав параметрам значения, соответствующие отсутствию ненужных компонентов. Например, если с помощью последовательной RLC-цепи нужно задать только резистор R, то надо задать 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 RsCs-цепи 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 резис-

тор R в RL- или RLC-цепь, не вводя новые компоненты в уже составленную схему, а просто задав их в окне параметров RLC-цепей.

Примеры моделирования RLC-цепей

Рисунок 15.13 показывает пример 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 за параметрами модели в целом.

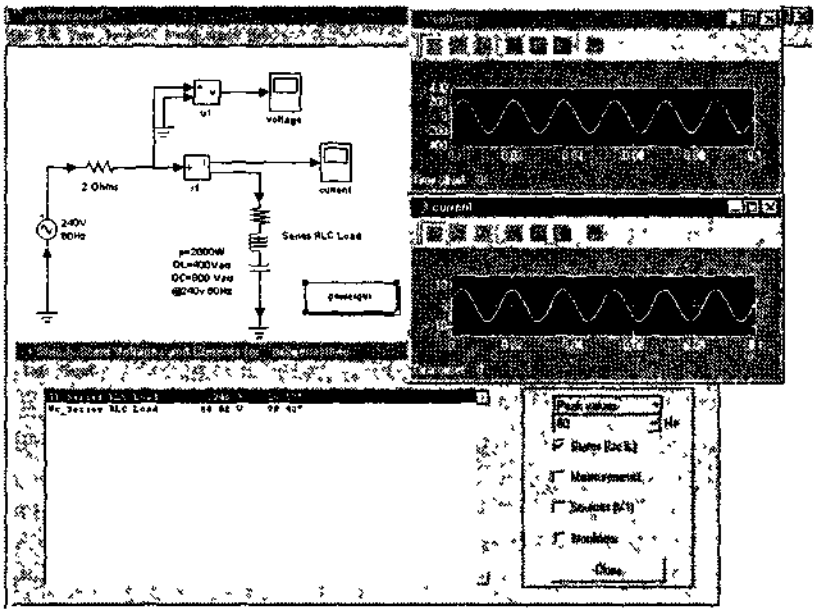


Рис. 15.13. Пример моделирования нагрузочной последовательной RLC-цепи, подключенной к источнику переменного напряжения

Активизация пиктограммы powergui выводит окно интерфейса (оно показано внизу рис. 15.13 вместе с окном вывода сообщений об ошибках). Окно powergui позволяет также контролировать состояние переменных модели и параметры ее компонентов. Более подробно работа с блоком powergui будет описана в дальнейшем.

На рис 15.14 представлен пример моделирования комбинированной цепи, которая образована конденсатором 4.42 мкФ и неполной параллельной RL-цепью Для исключения из параллельной цепи конденсатора C достаточно задать его емкость равной нулю.

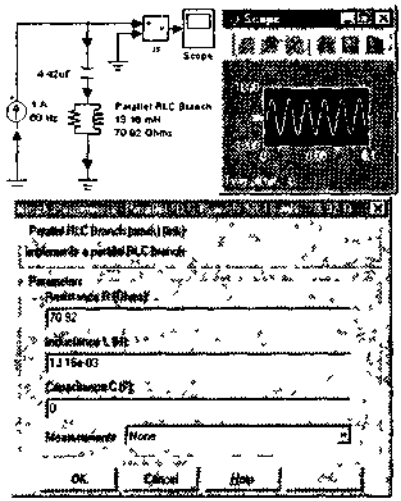


Рис. 15.14. Пример моделирования комбинированной цепи

В данном случае комбинированная цепь настроена на частоту резонанса $f_p = 1/2\pi\sqrt{LC}$ для 11 гармоники (660 Гц) переменного тока с частотой 60 Гц, источник которого нагружен данной цепью.

Контролируется напряжение 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 выходе цепи. В то же время напряжение на выходе цепи остается синусоидальным и имеет частоту источника 60 Гц, что является следствием линейности данной модели (как известно, в линейных моделях возникновение новых частот спектра исключено).

Модель линейного трансформатора

В пакете Power System Blockset имеется модель линейного трансформатора Linear Transformer, представленная на рис. 15.15. Она задается индуктивностью L_m и сопротивлением потерь в сердечнике R_m первичной обмотки трансформатора, а также омическими сопро-

тивлениями R_i и индуктивностями рассеяния L_i всех обмоток трансформатора ($i = 1, 2, 3$).

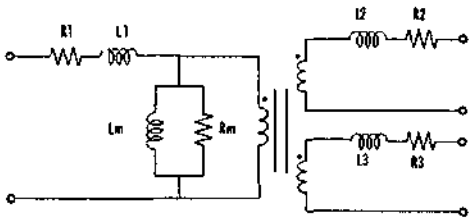


Рис. 15.15. Модель линейного трансформатора

При описании параметров трансформатора используется дополнительная система параметров, принятая в европейской промышленности и называемая в описании пакета pu-системой. При этом вводятся обозначения, представленные на рис. 15.16 (на рисунке также показаны примеры пересчета). Запись обозначений здесь не вполне корректна, так как при формальном математическом подходе означает $pu = R_{base} = L_{base}$, что является грубой ошибкой. На самом деле в системе pu-единиц под 1 pu понимаются разные параметры.

$$Rbase = 1\,pu = \frac{(Vn)^2}{Pn}$$

$$Lbase = 1\,pu = \frac{Rbase}{2\pi f_n}$$

$$Rbase = \frac{(424.35 \times 10^3)^2}{250 \times 10^6} = 720.3\Omega$$

$$Lbase = \frac{720.3}{2\pi 60} = 1.91H$$

$$R_1 = 0.002pu \times 720.3\Omega = 1.44\Omega$$

$$L_1 = 0.08pu \times 1.91H = 0.1528H$$

$$R_m = 500pu \times 720.3\Omega = 3.6 \times 10^5\Omega$$

Рис. 15.16. Формулы пересчета параметров R_{base} и L_{base} для трансформатора

Вид окна параметров линейного трансформатора представлен на рис. 15.17. Обратите внимание на то, что некоторые параметры трансформатора задаются списками, поскольку трансформатор может иметь несколько обмоток.

На рис. 15.18 дан пример моделирования трансформатора линейной подстанции с мощностью 75 кВт. Напряжение на входе трансфор-

матора 14 400 В, на выходе — 240 В и 2 × 120 В. Осциллографы контролируют ток в первичной обмотке и напряжение на одной из вторичных обмоток. Модель линейного трансформатора значительно идеализирована.

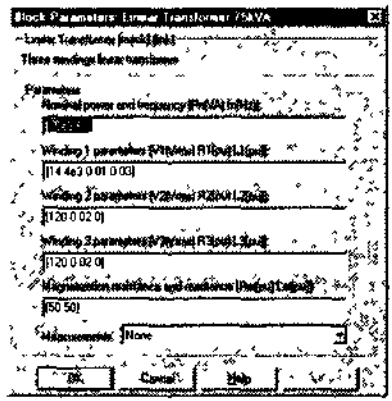


Рис. 15.17. Окно параметров линейного трансформатора

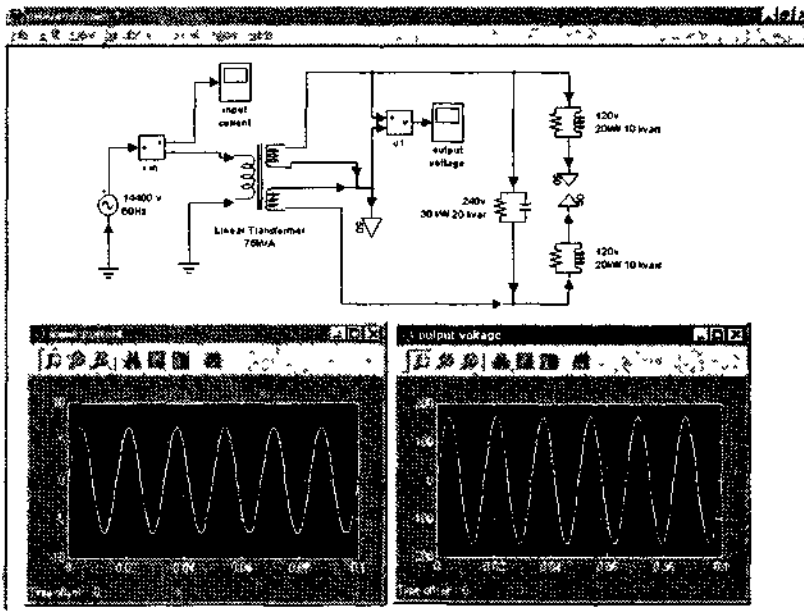


Рис. 15.18. Пример моделирования трансформатора линейной подстанции

Модель нелинейного трансформатора

Модель нелинейного трансформатора Saturable Transformer (рис. 15.19) отличается от модели линейного трансформатора тем, что индуктивность первичной обмотки L_m заменена на нелинейную индуктивность L_{sat}. Для нелинейных трансформаторов также может использоваться pu-система параметров (рис. 15.16).

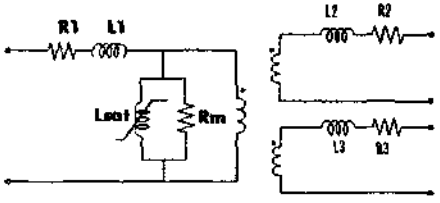


Рис. 15.19. Модель нелинейного трансформатора

Нелинейность трансформатора учитывается зависимостями, представленными на рис. 15.20. Допускаются два вида этой зависимости, отличающиеся числом опорных точек и поведением зависимости в области малых токов.

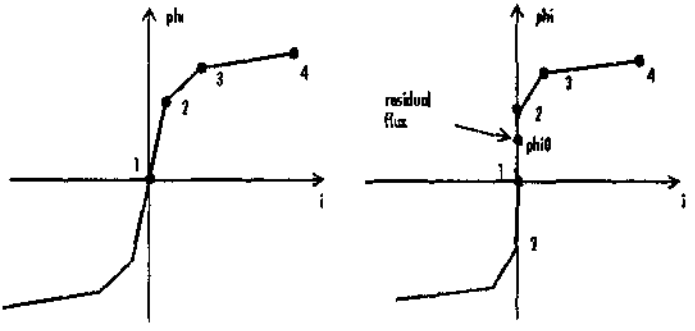


Рис. 15.20. Модели нелинейности трансформатора

На рис. 15.21 представлен пример моделирования одной фазы мощной трансформаторной подстанции с установочной мощностью 1000 МВт. Там же показано окно параметров нелинейного трансформатора.

Нетрудно заметить (рис. 15.22), что переходный процесс в системе с нелинейным трансформатором намного сложнее, чем в системе с линейным трансформатором. При этом он длится примерно два десятка периодов переменного напряжения от источника электропитания на входе системы. Эти переходные процессы получены с помощью блока мультиметра Multimeter и виртуального многоканального осциллографа Scope.

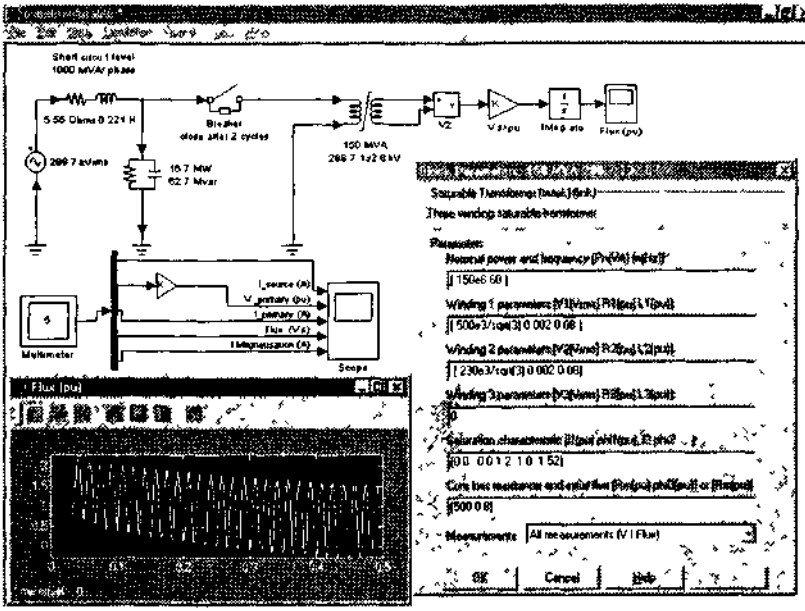


Рис. 15.21. Пример моделирования фазы трансформаторной подстанции

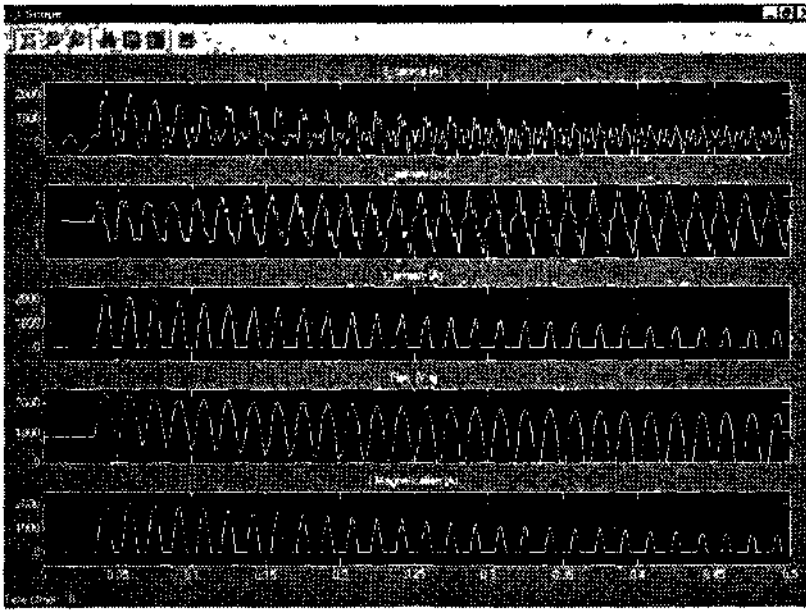


Рис. 15.22. Результаты моделирования системы с мощным нелинейным трансформатором, полученные с помощью блока измерений

К сожалению, модель нелинейного трансформатора не учитывает гистерезиса намагничивания сердечника и потому является лишь первым приближением к учету явлений, связанных с нелинейностью трансформатора Однако пользователь может создать свою модель трансформатора, в том числе учитывающую явление гистерезиса.

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 взаимной индуктивности Mutual Inductance на основе идеального трансформатора (рис. 15.23). Он соответствует теоретической модели взаимной индуктивности.

Пример моделирования электрической цепи с блоком взаимной индуктивности представлен на рис. 15.24. Там же представлено окно параметров этого блока. Параметры обмоток трансформатора, имитирующего взаимную индуктивность, задаются списками.

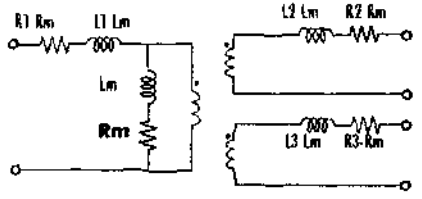


Рис. 15.23. Блок взаимной индуктивности

В этом примере создается двухчастотное напряжение на RL-нагрузке, причем первая (60 Гц) и третья (180 Гц) гармоники создаются благодаря взаимной индуктивности, представленной блоком Mutual Inductance.

Нелинейный ограничитель пиковых напряжений

В пакете Power System Blockset можно задавать ограничители пиковых напряжений Surge Arrester (варисторы) с вольтамперной характеристикой, которая описывается следующим выражением:

$$\frac{V}{Vref} = K_i \left(\frac{I}{Iref} \right)^{1/\alpha_i}$$

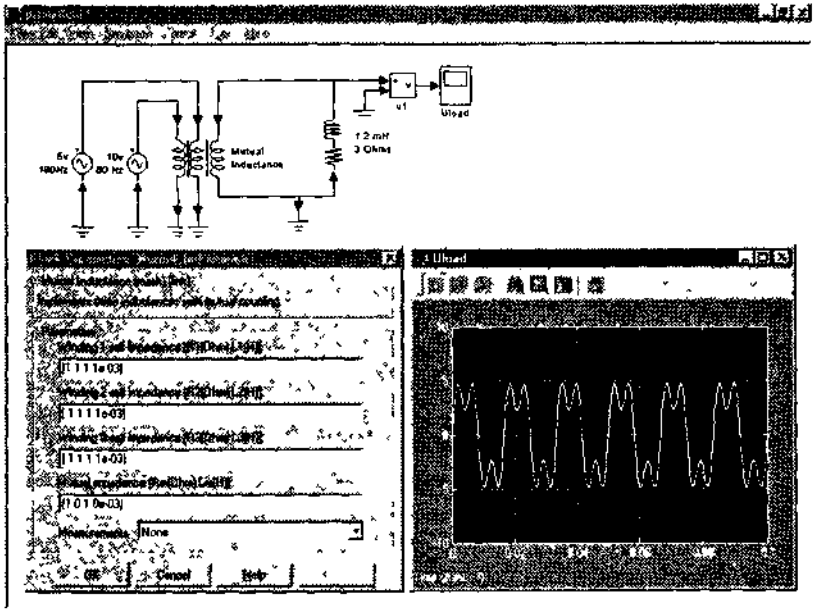


Рис. 15.24. Пример моделирования электрической цепи с блоком взаимной индуктивности

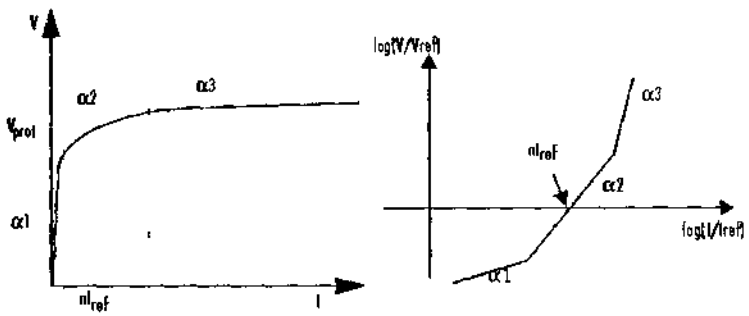


Рис. 15.25. Вольтамперные характеристики нелинейного ограничителя в *линейном* (слева) и в *логарифмическом* (справа) масштабах

Такую вольтамперную характеристику можно представить в виде, показанном на рис. 15.25, — слева в обычном масштабе и справа в логарифмическом. На этом рисунке четко видна возможность разбиения вольтамперной характеристики на несколько характерных участков, что и используется при ее задании в окне параметров ограничителя.

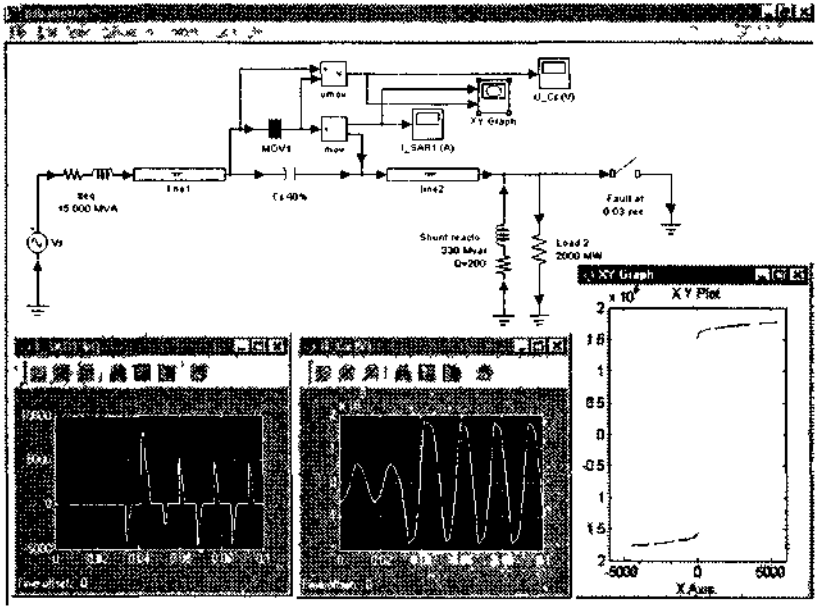


Рис. 15.26. Моделирование электрической системы с варистором — ограничителем выбросов напряжения

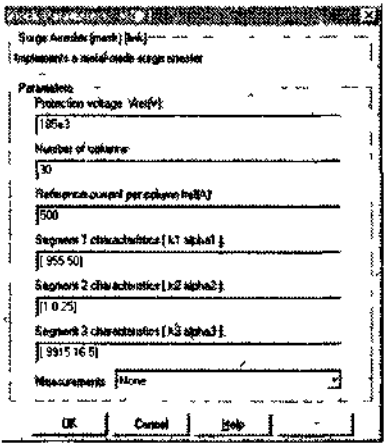


Рис. 15.27. Окно установки параметров варистора

На рис. 15.26 представлена модель электрической схемы с варистором и результаты моделирования. Обратите внимание на то, что

помимо осциллограмм напряжений, иллюстрирующих эффективное ограничение выбросов, представлено окно графопостроителя с вольтамперной характеристикой варистора. Окно параметров варистора представлено на рис. 15.27.

Применение варисторов является одним из самых простых и эффективных методов ограничения выбросов напряжения. Наряду с мощными варисторами 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 (в том числе с помощью пакета Power System Blockset) особенно ценным.

Выключатель

Для имитации выключателей (рубильников) служит управляемый выключатель Breaker. Он обеспечивает включение или выключение переменного тока. Пример применения этого элемента дан на рис. 15.28. Там же показано окно параметров этого блока и результаты моделирования.

Обратите внимание, что возможно задание паразитных сопротивления и индуктивности выключателя, что позволяет (при необходимости) моделировать эффекты, связанные с неидеальностью выключателя.

Линии передачи с сосредоточенными параметрами

В пакете Power System Blockset имеется возможность задания двух типов линий передачи. Первый тип — это линия с сосредоточенными параметрами PI Section Line, структура которых представлена на рис. 15.29. Она может приближенно представлять и линию с распределенными постоянными. В последнем случае длина линий задается в километрах и параметрами линий являются распределенные индуктивности и емкости, измеряемые соответственно в Гн/км и Ф/км. Взаимная индуктивность у таких линий не учитывается.

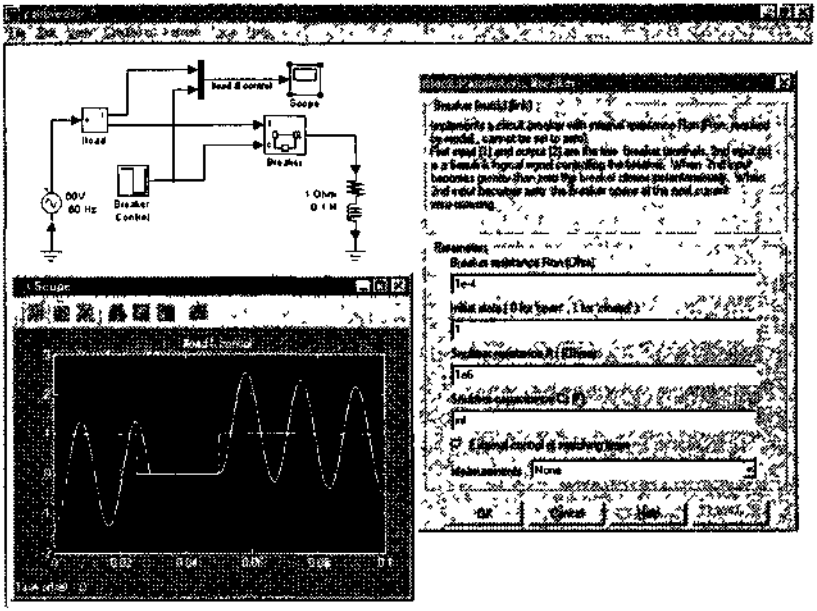


Рис. 15.28. Пример моделирования цепи с выключателем

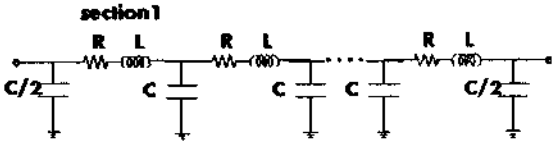


Рис. 15.29. Линия типа PI Section Line

Максимальная частота переменного тока, который может протекать через линию данного типа, составляет $f_{max} = N_c/8l$, где N — число секций линии, $v = (LC)^{-1}$ — 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. 15.30 показана модель электрической схемы с линией передачи рассматриваемого типа. Там же показано окно 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.

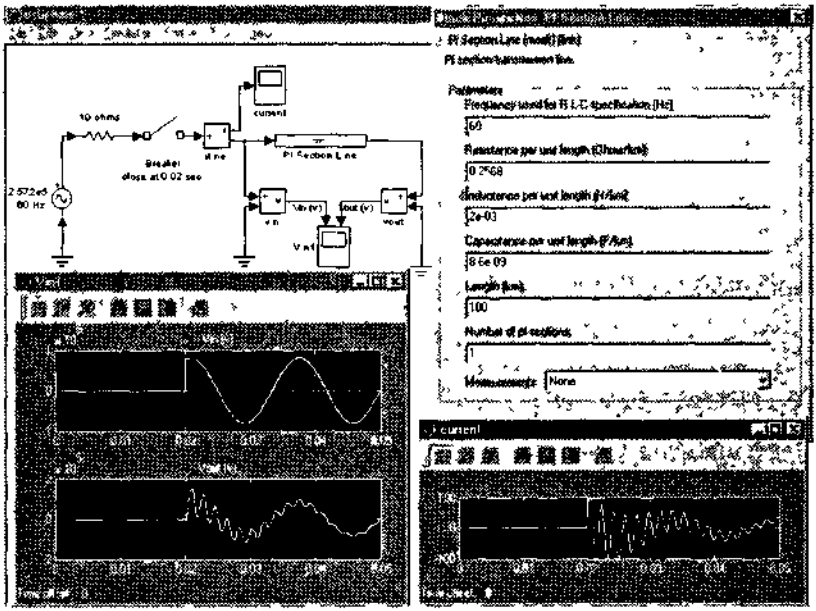


Рис. 15.30. Пример моделирования электрической схемы с линией типа PI Section Line

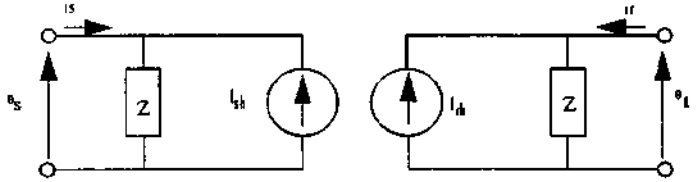
Полезно обратить внимание на довольно сложные переходные процессы, возникающие при включении источника переменного напряжения на линию. Помимо переходного процесса, происходящего с частотой этого источника, виден более высокочастотный переходной процесс. Он особенно заметен на осциллограмме линии.

Линии передачи с распределенными параметрами

Другой тип линий — линии с распределенными параметрами Distributed Parameters Line. Теоретическая модель такой линии с соответствующими расчетными выражениями представлена на рис. 15.31. Смысл этих выражений очевиден для специалистов, применяющих подобные линии, и потому здесь детально не обсуждается.

Пример использования линии передачи с распределенными параметрами представлен на рис. 15.32. Там же даны осциллограммы результата моделирования. В этом примере моделируется процесс включения источника переменного напряжения с частотой 60 Гц на

отрезке линии длиной 300 км. Окно установки параметров распределенной линии также представлено на рис. 15.32: задается число секций линии, частота работы линии, значение распределенной индуктивности и емкости и длина линии.



$$I_{sh}(t) = \left(\frac{1+h}{2}\right)\left[\frac{1}{Z}e_r(t-\tau)+h\,i_r(t-\tau)\right]+\left(\frac{1-h}{2}\right)\left[\frac{1}{Z}e_s(t-\tau)+h\,i_s(t-\tau)\right]$$

$$I_{rh}(t) = \left(\frac{1+h}{2}\right)\left[\frac{1}{Z}e_s(t-\tau)+h\,i_s(t-\tau)\right]+\left(\frac{1-h}{2}\right)\left[\frac{1}{Z}e_r(t-\tau)+h\,i_r(t-\tau)\right]$$

$$Z = Z_C+\frac{R}{4} \qquad h = \frac{Z_C-\frac{R}{4}}{Z_C+\frac{R}{4}} \qquad Z_C = \sqrt{\frac{L}{C}} \qquad \tau = d\sqrt{LC}$$

Рис. 15.31. Теоретическая модель линии с распределенными параметрами

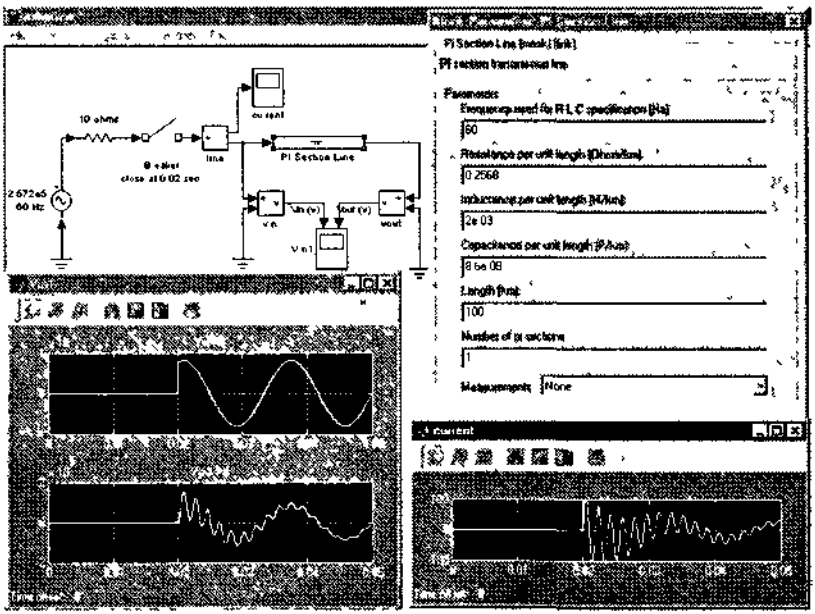


Рис. 15.32. Пример использования линии с распределенными параметрами

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 резистором 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.

Коммутирующие элементы энергетической электроники

Состав библиотеки энергетической электроники

Современная силовая электроника основана на импульсном способе преобразования электрической энергии, обеспечивающем высокий коэффициент полезного действия преобразовательных устройств.

Поэтому в библиотеку рассматриваемого пакета включен достаточно представительный набор блоков коммутирующих устройств. Активизация пиктограммы Power Electronics открывает окно с пиктограммами моделей управляемых ключей (рис. 15.33).

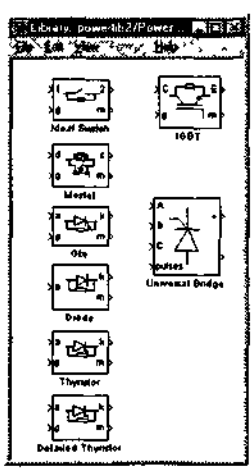


Рис. 15.33. Состав библиотеки компонентов энергетической электроники

Представлены следующие типы ключей:

- Ideal Swith — идеальный управляемый ключ;
- Mosfet — полевой транзистор с изолированным затвором;
- Gto — запираемый тиристор (Gate turn off);
- Diode — полупроводниковый диод;
- Thyristor — упрощенная модель тиристора;
- Detailed thyristor — уточненная модель тиристора;
- IGBT — силовой биполярно-полевой модуль типа JGBT;
- Universal Bridge — универсальный модуль моста.

Два последних блока появились в версии 2.0 данного пакета. Некоторое удивление может вызвать отсутствие модели биполярного транзистора. Отчасти это связано с тем, что в последнее время силовые биполярные транзисторы вытесняются полевыми транзисторами, а отчасти и тем, что пользователь (в случае необходимости) может составить подсистему биполярного транзистора, например на основе модели идеального ключа.

Модели коммутирующих устройств

Все модели коммутирующих элементов содержат гасящую выбросы напряжения последовательную RsCs-цепь, которая подключается к силовым выводам моделей. Задание бесконечного значения C_s и нулевого R_s закорачивает модель (пиктограмма устройства при этом заменяется пиктограммой проводника). Модели имеют также выход m для подключения измерительных приборов. На этом выходе формируется список значений тока, протекающего через устройство, и напряжения на нем в процессе моделирования, что позволяет строить системы, управляемые этими параметрами.

Управляемый ключ

Идеальный ключ Ideal Swith моделирует ключ, который во включенном состоянии имеет сопротивление R_{on} и индуктивность L_{on}. Сопротивление R_{on} позволяет приближенно учитывать статические потери в ключе во включенном состоянии, а индуктивность L_{on} — инерционные процессы при переключении. Задание $L_{on}=0$ недопустимо, поскольку ведет к неразрешимости системы уравнений, описывающих работу электрических цепей, из-за деления на ноль. Сопротив-

ление ключа в выключенном состоянии считается равным бесконечности. Если требуется задать конечное сопротивление, 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, можно использовать последовательную RsCs-цепь, которая включается параллельно ключу для ограничения выбросов напряжения и подавления дуги. Окно задания параметров ключа и пример моделирования цепи с ключом представлены на рис 15.34.

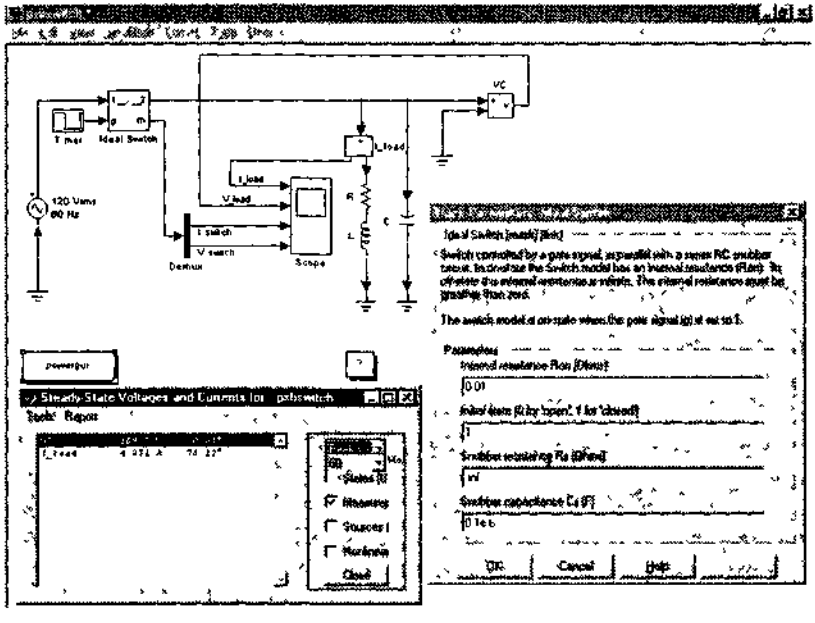


Рис. 15.34. Пример моделирования цепи с идеальным ключом

Осциллограммы на рис. 15.35 иллюстрируют процесс отключения RLC-контура от источника переменного напряжения 120 В 60 Гц. Нетрудно заметить, что отключение сопровождается явно выраженным затухающим высокочастотным переходным процессом.

Данный пример моделирования показывает, что название «идеальный ключ» (в оригинале Ideal Switch) является неточным и сужает возможности модели. На самом деле данная модель описывает реальный ключ с его основными паразитными параметрами. Они и

создают заметные высокочастотные колебательные процессы, наблюдаемые при моделировании цепей с такими ключами.

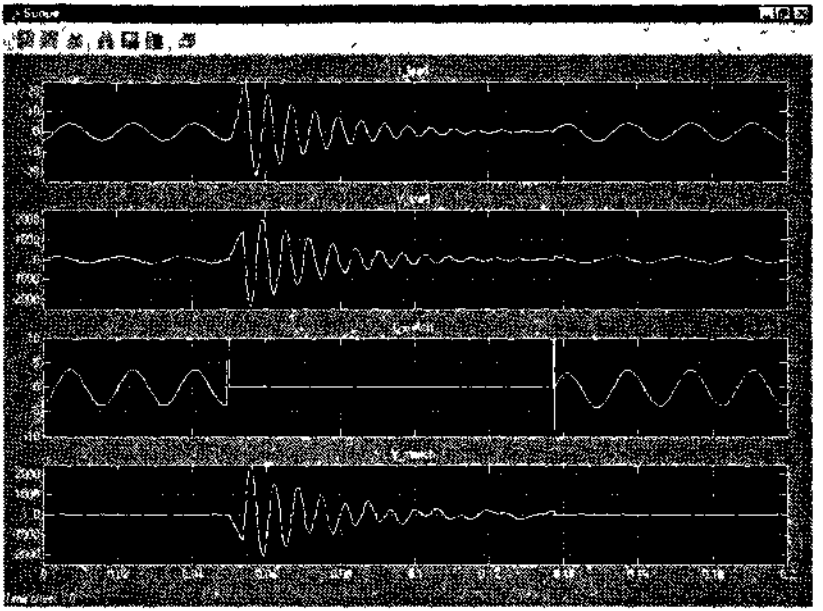


Рис. 15.35. Переходные процессы при отключении RLC-контура от источника переменного напряжения

Полупроводниковый диод

Модель диода Diode представляет собой последовательно соединенные источник напряжения V_f на диоде в прямом включении, резистор R_{on} и паразитную индуктивность L_{on} в прямом направлении, когда диод проводит ток. В обратном направлении сопротивление диода считается бесконечно большим. Предусмотрен учет включения параллельно диоду последовательной гасящей цепи RsCs (эта цепь является внутренней и в составе модели не показана). Все эти возможности дает окно установки параметров диода, представленное на рис. 15 36 (справа)

Рисунок 15 36 содержит пример моделирования однополупериодного диодного выпрямителя, работающего на RL-нагрузку с небольшой индуктивностью Большое количество осциллограмм позволяет достаточно полно судить о физических процессах, протекающих в данной схеме, а точнее о результатах их моделирования.

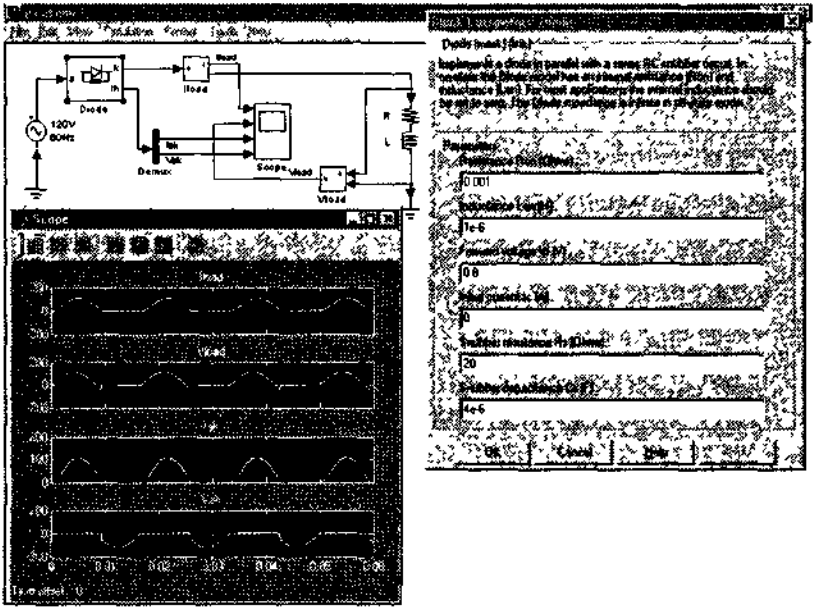


Рис. 15.36. Пример моделирования однополупериодного диодного выпрямителя

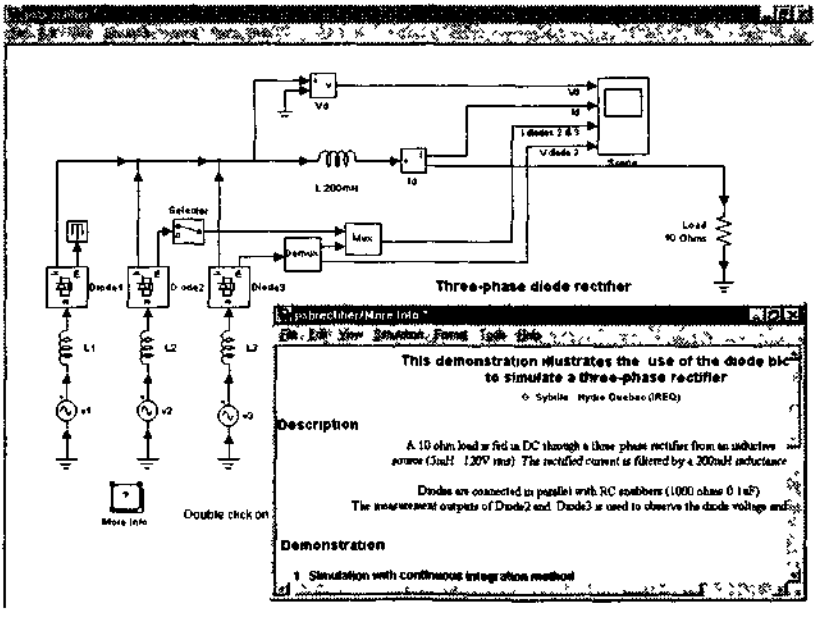


Рис. 15.37. Пример моделирования трехфазного выпрямителя

Следует отметить, что модель диода довольно приближенная. Так, источник напряжения V_f дает ненулевое напряжение на диоде при нулевом токе через диод. У реального диода это напряжение равно нулю. Лишь приближенно прямая ветвь диода может описываться прямой, наклон которой задается номиналом резистора R_{on} (в действительности ток диода является экспоненциальной зависимостью от напряжения). Наконец, не полностью учитывается инерционность диода, связанная с накоплением в его базе избыточных зарядов неосновных носителей и их рассасыванием при выключении. Учет этих явлений индуктивностью L_{on} является довольно грубым приближением. Тем не менее принятые приближения вполне соответствуют тем, что применяются в практике использования диодов при проектировании силовых устройств.

Другой пример (рис. 15.37) 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 установившимся напряжением около 12,5 В и довольно небольшими пульсациями. Последнее является следствием выпрямления всех трех фаз, сдвинутых на угол 120° относительно друг друга.

Осциллограммы, детально иллюстрирующие работу последнего примера, полученные с помощью виртуального многоканального осциллографа, приведены на рис. 15.38.

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).

Полевой транзистор с изолированным затвором

Полевые транзисторы с изолированным затвором в последнее время стали основным типом переключающих транзисторов малой и средней (а иногда и большой) мощности. В пакете Power System Blockset предусмотрена простая модель полевого транзистора Mosfet. Фактически он рассматривается как силовой ключ с сопротивлением R_{on} и индуктивностью L_{on} во включенном состоянии и бесконечно большим сопротивлением в выключенном состоянии. Можно также задать включение параллельно транзистору диода, открытого при закрытом транзисторе и характеризующегося сопротивлением R_d.

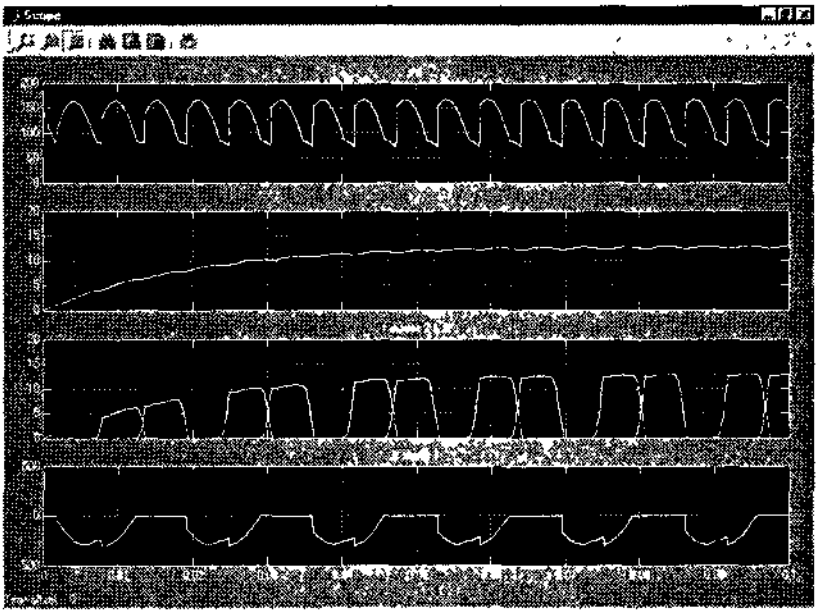


Рис. 15.38. Осциллограммы работы трехфазного выпрямителя

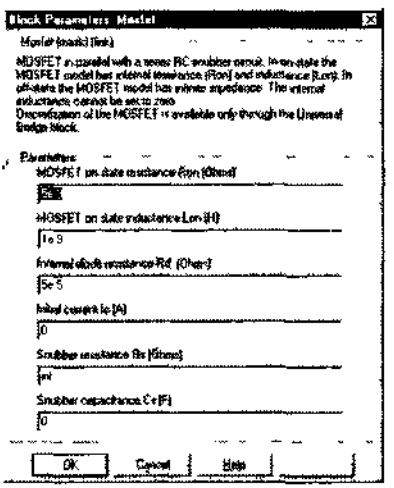


Рис. 15.39. Окно параметров полевого транзистора

Наконец, можно добавить подключенную к выводам сток-исток последовательную RsCs-цепь Все эти параметры устанавливаются в окне параметров, представленном на рис 15.39.

На рис. 15.40 дан пример моделирования квазирезонансного преобразователя на полевом транзисторе, переключающемся в моменты прохождения тока через ноль. Обратите внимание на фазовый портрет процессов работы преобразователя, полученный с помощью блока графопостроителя.

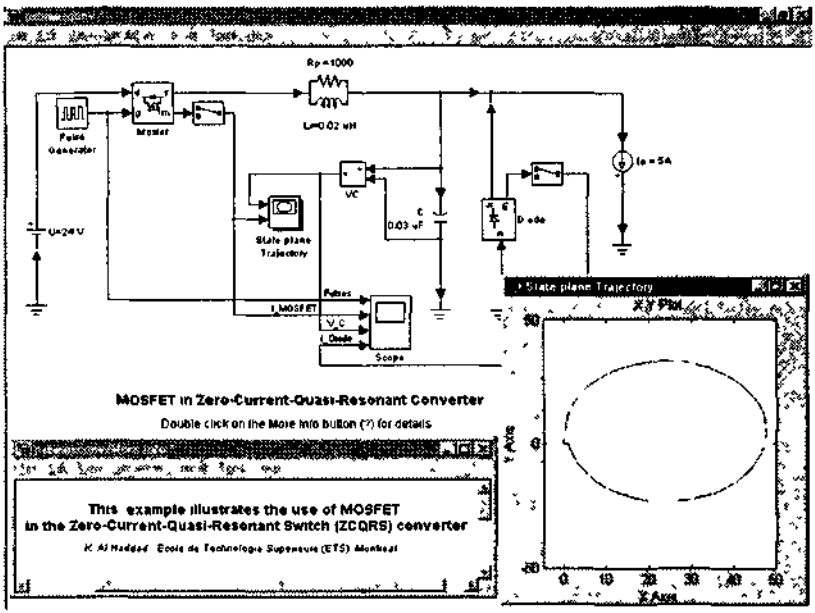


Рис. 15.40. Моделирование квазирезонансного преобразователя на полевом транзисторе

Переключение коммутирующего устройства (в нашем случае полевого транзистора) в моменты перехода тока в колебательном контуре через ноль минимизирует динамические потери в ключе и позволяет получить приемлемый коэффициент полезного действия на высоких частотах преобразования — вплоть до сотен кГц и даже нескольких МГц Осциллограммы работы преобразователя представлены на рис 15.41.

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.

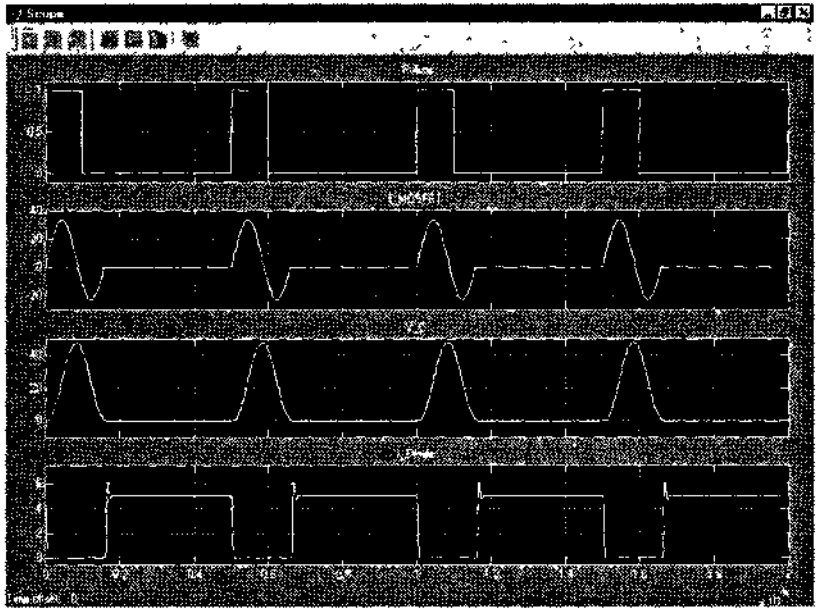


Рис. 15.41. Осциллограммы работы квазирезонансного преобразователя на полевом транзисторе

Упрощенная модель тиристора

Модель тиристора Thyristor также построена на основе идеального ключа с элементами, имитирующими остаточные параметры включенного тиристора. Это сопротивление во включенном состоянии R_{on}, индуктивность L_{on} и падение напряжения в прямом направлении V_f. В выключенном состоянии (обратное направление) сопротивление устройства считается равным бесконечности. Тиристор выключается, если управляющий сигнал равен нулю, а также в тех случаях, когда прямой ток тиристора падает до нуля или напряжение на тиристоре достигает значения обратного напряжения. Предусмотрено также параллельное включение (между анодом и катодом) последовательной RsCs-цепи. Можно также задавать I_c, с которого начинается моделирование (по умолчанию 0, то есть моделирование начинается при закрытом тиристоре).

Окно параметров упрощенной модели тиристора и осциллограммы ее работы показаны на рис. 15.42. Задание $L_{on} = 0$ недопустимо. Там же представлен пример схемы с тиристором. Обратите внимание, что модель тиристора имеет надпись Thyristor, что указывает на применение упрощенной модели тиристора. Выполняется моделирование тиристора со следующими параметрами: R = 1 Ом; L = 10 мГн; $R_{on} = 0.001$ Ом, $L_{on} = 1 \cdot 10^{-5}$ Гн, $V_f = 0.8$ В, $I_c = 0$ А, $R_s = 20$ Ом, $C_s = 4 \cdot 10^{-6}$ Ф. Параметры моделирования: решатель дифференциальных уравнений ode15s; относительная погрешность Relative tolerance $1 \cdot 10^{-3}$; абсолютная погрешность Absolute tolerance $1 \cdot 10^{-3}$.

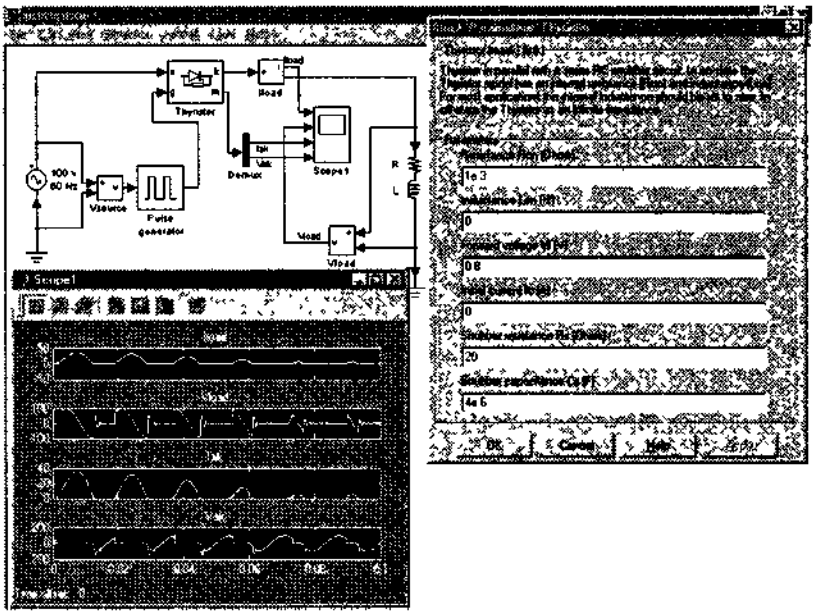


Рис. 15.42. Пример моделирования цепи с тиристором

Уточненная модель тиристора

Имеется более детальная модель тиристора — Detail Thyristor. Окно параметров этой модели имеет два дополнительных параметра: ток I_l, при котором тиристор выключается, и время выключения T_q (в секундах), которое характеризует задержку выключения. Эта модель позволяет более точно моделировать переходные процессы в схе-

мах 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

Запираемый тиристор Gto

Запираемые тиристоры Gto — сравнительно новый и перспективный тип мощных силовых коммутирующих элементов. Они имеют управляющие сигналы малой мощности и способны переключать большие напряжения и токи, чем полевые транзисторы с изолированным затвором. Однако для этих устройств характерно значительное время выключения. В последнее время они вытесняются модулями типа IGBT, описанными ниже.

Кроме набора параметров, общих с обычными тиристорами, запираемые тиристоры имеют два новых специфических параметра: время спада тока до уровня 0.1 от тока в момент выключения (T_f) и время окончательного спада тока до нуля (T_t). Временная зависимость спада тока приближенно описывается двумя линейными участками с указанными длительностями.

На рис. 15.43 показана схема преобразователя постоянного напряжения, в котором используется Gto-модуль. Осциллограммы поясняют работу этой схемы. Нагрузкой преобразователя является RL-цепь и источник постоянного напряжения E (первичным источником является источник напряжения U).

Gto-модули 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.

Силовой модуль IGBT

Силовой модуль IGBT (рис. 15.44) — новый перспективный элемент энергетической электроники. Он создан на основе комбинации биполярных транзисторов с полевыми. В прежних версиях описываемого пакета блок IGBT-модуля отсутствовал.

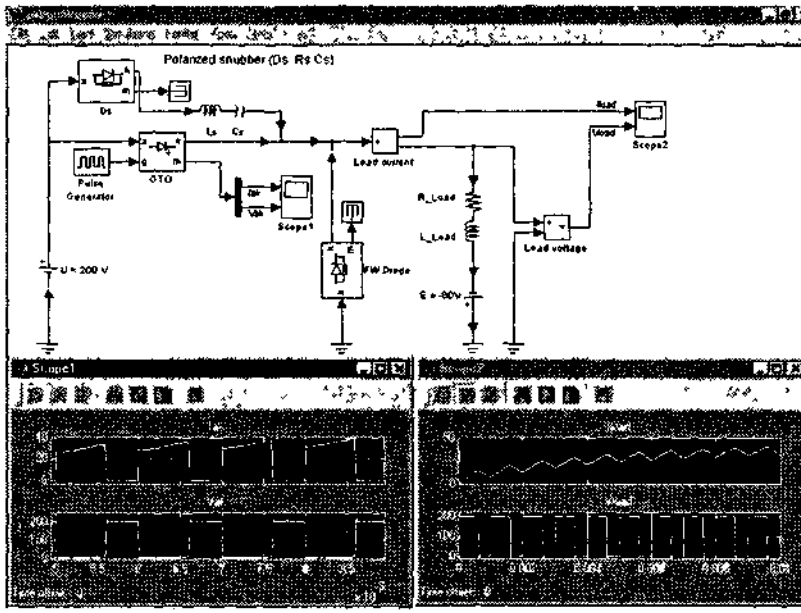


Рис. 15.43. Моделирование преобразователя постоянного напряжения с Gto-модулем

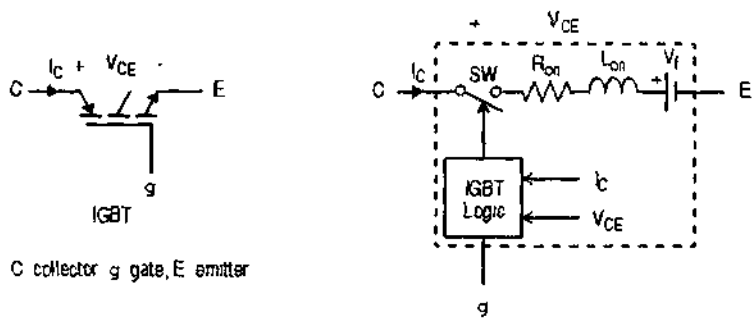


Рис. 15.44. Блок IGBT: обозначение (слева) и модель (справа)

Статическая вольтамперная характеристика модуля может быть представлена в виде двух отрезков прямых (рис. 15.45). Горизонтальный участок характерен для выключенного состояния модуля, наклонный — для включенного состояния. Наклон последнего задается сопротивлением устройства во включенном состоянии (оно очень мало). Остаточное напряжение при малых токах, протекающих через устройство, учитывается параметром V_f.

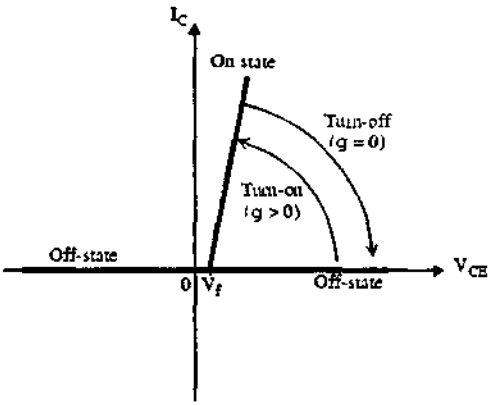


Рис. 15.45. Статическая вольтамперная характеристика модуля IGBT

Для устройств IGBT, как и для GTO, характерен довольно медленный процесс выключения, состоящий из двух стадий. Его временные диаграммы, представленные на рис. 15.46, демонстрируют два важных временных параметра блока JGBT — время спада тока T_f и T_t.

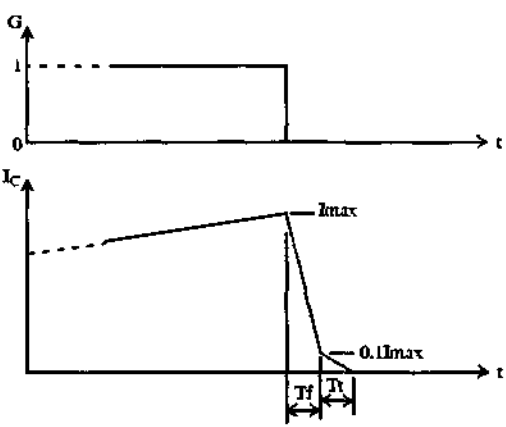


Рис. 15.46. Временные диаграммы выключения модуля IGBT

Окно установки параметров блока IGBT представлено на рис. 15.47. Все они уже описаны выше.

На рис. 15.48 представлен пример моделирования повышающего напряжение импульсного преобразователя на основе IGBT-модуля. Это типичный обратноходовой преобразователь с индуктивным накопителем энергии.

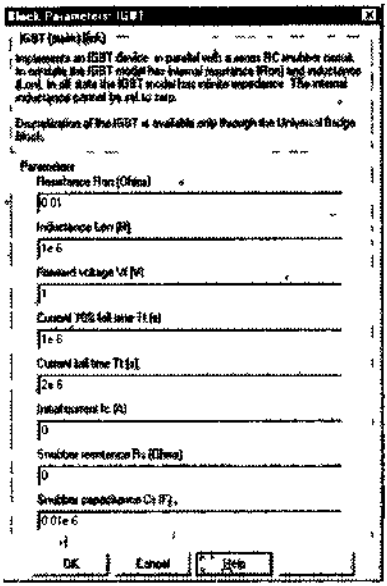


Рис. 15.47. Окно установки параметров блока IGBT

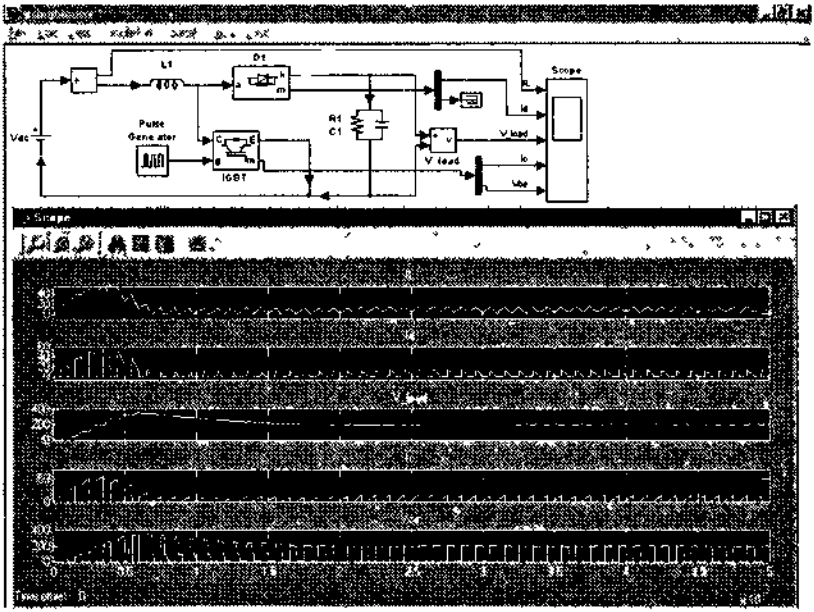


Рис. 15.48. Повышающий напряжение импульсный преобразователь на основе IGBT-модуля

Приведенные на рис. 15.48 осциллограммы показывают наличие довольно длительного переходного процесса, в ходе которого наблюдается выброс выходного напряжения, почти вдвое превышающий установившееся значение.

Универсальный мостовой модуль

Благодаря широкому применению мостовых схем в новую версию пакета Power System Blockset был включен универсальный мостовой модуль Universal Bridge. На рис. 15.49 приведены два варианта диодного моста. Однако такой модуль может быть создан и на основе других описанных выше силовых коммутирующих элементов.

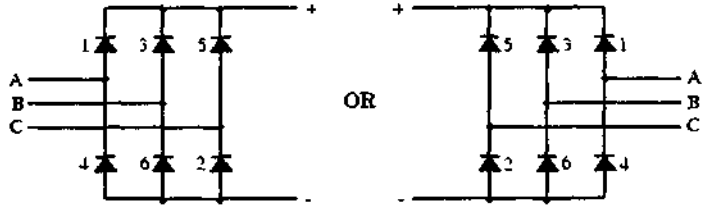


Рис. 15.49. Варианты схемы диодного моста Universal Bridge

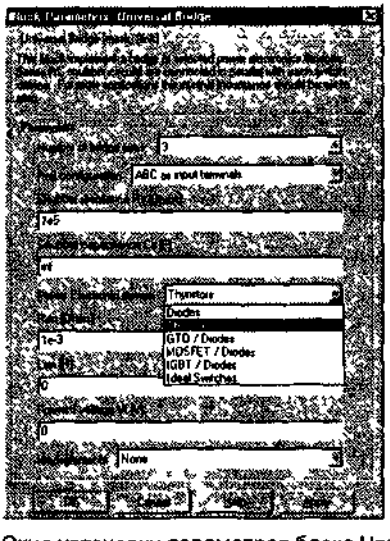


Рис. 15.50. Окно установки параметров блока Universal Bridge

Окно параметров блока Universal Bridge представлено на рис. 14.50 Большинство параметров вполне очевидны и соответствуют описанным выше параметрам силовых устройств Необходимо отметить раскрывающийся список элементов, на основе которых строится мост (на рис 15 50 этот список открыт). Кроме того, можно задавать число элементов моста — параметр Number of bridge arms (1, 2 или 3), а также конфигурацию моста — параметр Port configuration (см варианты на рис 15.39)

На рис. 15 51 дан пример моделирования преобразователя переменного напряжения в постоянное, которое затем вновь преобразуется в переменное напряжение.

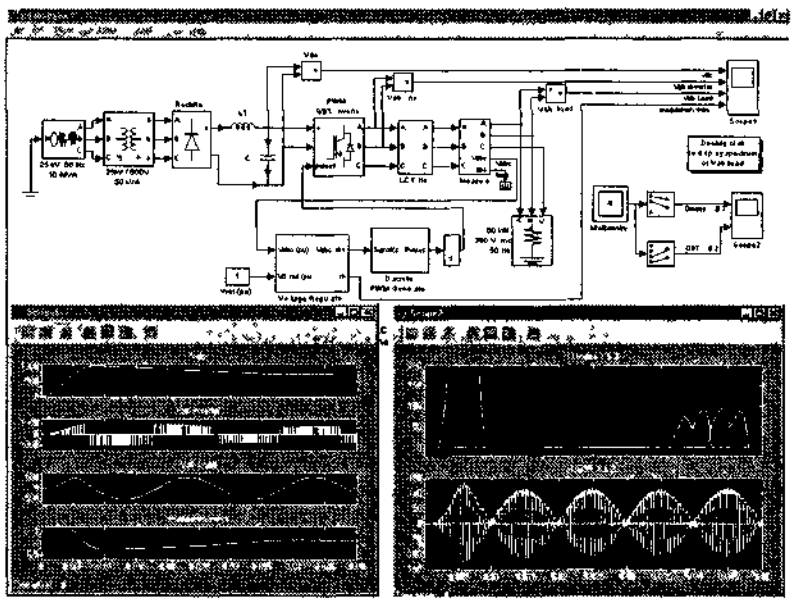


Рис. 15.51. Пример преобразователя на основе блока Universal Bridge

Еще одна интересная деталь этого примера — возможность анализа спектра выходного сигнала преобразователя Окно анализатора спектра (рис. 15.52) вызывается нажатием пиктограммы, расположенной справа между блоками осциллографов

В этом окне сверху показана кривая выходного напряжения преобразователя, на которой отчетливо видны малые по амплитуде колебания, обусловленные процессами импульсного преобразования.

Внизу показан спектр выходного сигнала Он позволяет уяснить допустимость высокочастотных пульсаций и продумать меры по их устранению

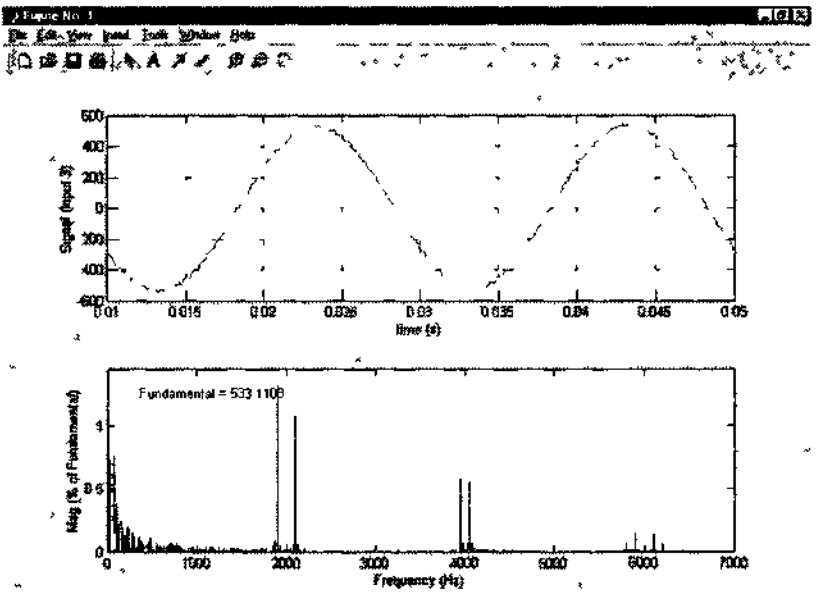


Рис. 15.52. Окно анализатора спектра

15

Моделирование электрических машин и схем управления ими

Как уже отмечалось, библиотека пакета Power System Blockset содержит модели ряда электрических машин постоянного тока, а также синхронных и асинхронных машин переменного тока Машины могут моделироваться в режимах двигателя 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 двигателя постоянного тока со стартером

Рисунок 15 53 показывает пример моделирования двигателя 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

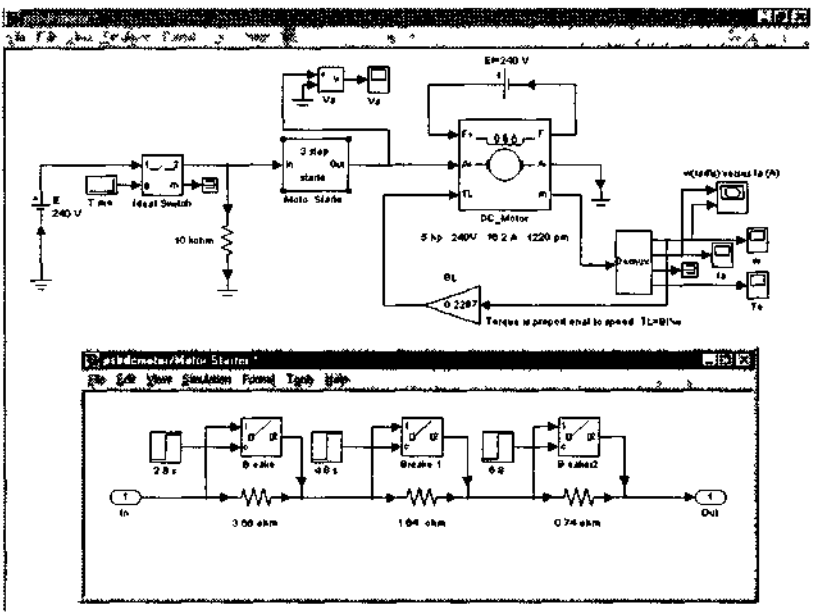


Рис. 15.53. Моделирование двигателя постоянного тока с трехшаговым стартером

Трехшаговый стартер не является стандартной моделью пакета Power System Blockset — он существует как подсистема Для вызова окна подсистемы (рис 15 53) и ее редактирования следует выполнить двойной щелчок мышью на пиктограмме подсистемы (в нашем случае Motor Starter) Как нетрудно заметить, стартер представляет собой три коммутируемых с помощью идеальных ключей резистора, включенных последовательно в цепь двигателя На практике для

их коммутации используются обычные реле подходящей мощности.

Несмотря на простоту моделируемой схемы, переходные процессы в ней достаточно сложны (рис. 15.54). Это типичный пример моделирования системы с переменными во времени параметрами. Нетрудно заметить, что система запуска в данном случае обеспечивает быстрый и монотонный набор заданного числа оборотов — см. осциллограмму зависимости числа оборотов w от времени на рис 15.54.

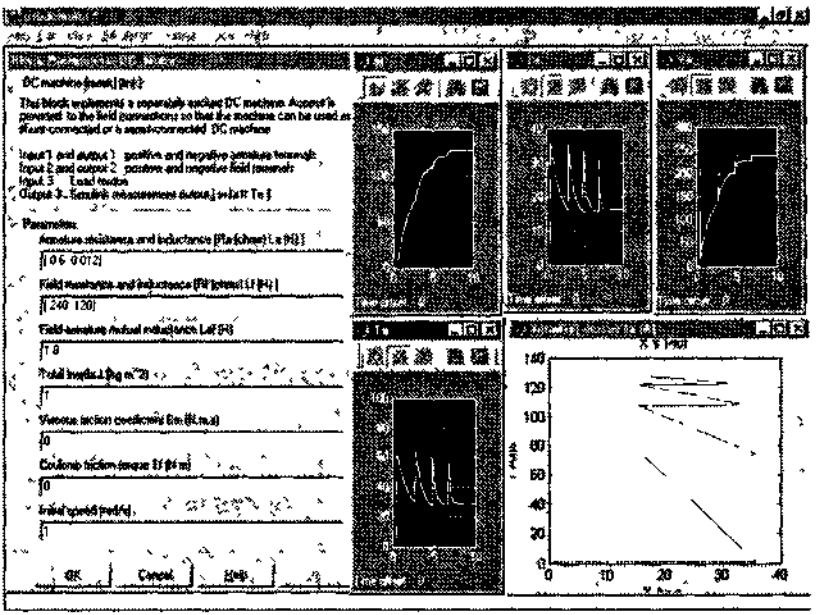


Рис. 15.54. Результаты моделирования двигателя постоянного тока с трехшаговым стартером

На рис. 15.54 в левой части представлено окно параметров двигателя постоянного тока.

Предоставим читателям, знакомым с машинами постоянного тока, возможность самостоятельно разобраться с параметрами таких машин, имеющихся в данном окне.

В демонстрационном примере psbdcdrive можно найти пример моделирования электропривода двигателя постоянного тока.

Моделирование синхронных машин (генераторов)

На рис. 15.55 дан пример моделирования синхронной машины. В этой модели используется ряд подсистем, показанных также на рис. 15.55. Таким образом, данная модель является многоуровневой системой.

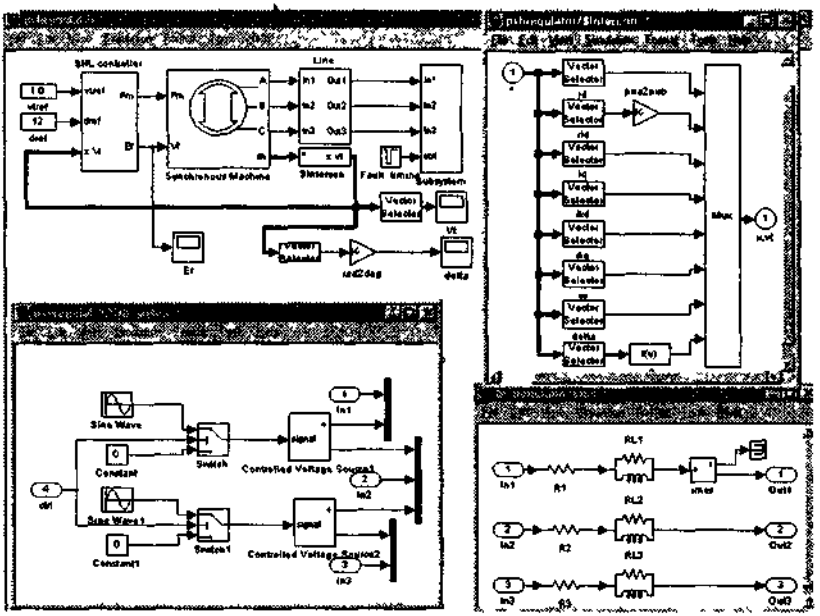


Рис. 15.55. Многоуровневая модель привода синхронной машины

На рис. 15.56 показана та же модель с открытым окном установки параметров синхронной машины и осциллограммами ее работы.

Моделирование мощной синхронной машины гидравлической турбины

На рис. 15.57 дан 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.

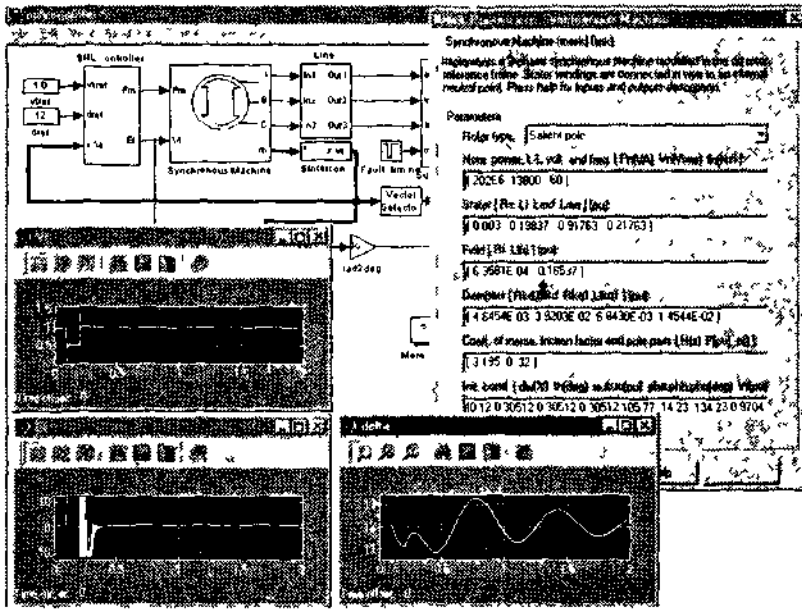


Рис. 15.56. Окно параметров синхронной машины и результаты моделирования

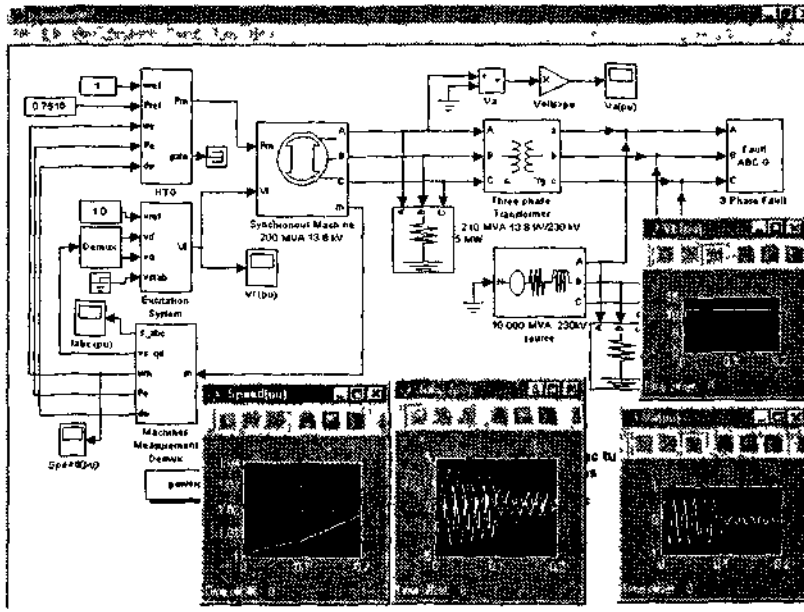


Рис. 15.57. Пример моделирования мощной синхронной машины

Рисунок 15.57 содержит также осциллограммы работы синхронной машины.

Моделирование синхронной машины (двигателя)

Еще один пример моделирования синхронной машины — двигателя с магнитом — представлен на рис. 15.58. Осциллограммы изменения во времени ряда параметров системы показывают весьма сложный характер переходных процессов.

Подсистемы этой машины и окно установки ее параметров представлены на рис. 15.59.

Приведенные примеры наглядно показывают возможности моделирования с помощью пакета Power System Blockset синхронных машин различного типа и систем привода, что открывает возможности детального изучения этих сложных устройств без привлечения к этому натурных испытаний или по крайней мере с ограничением их применения.

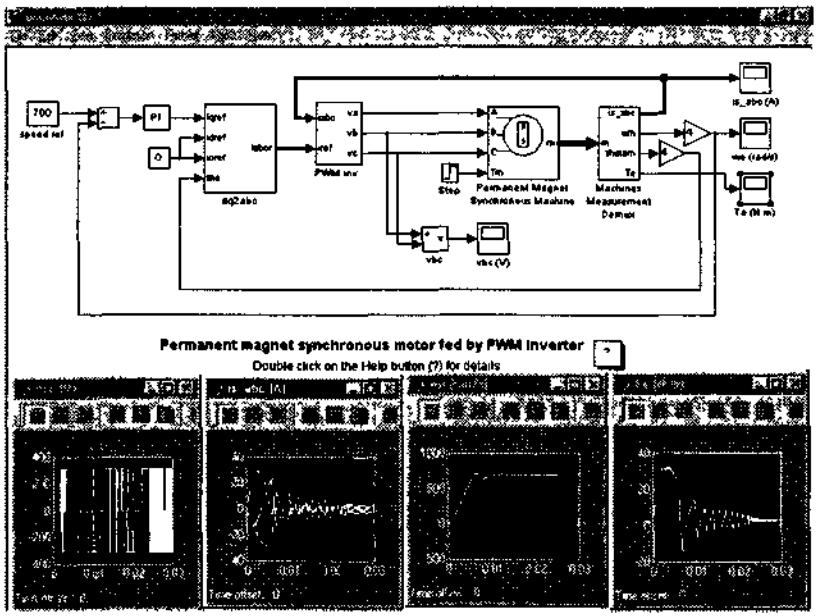


Рис. 15.58. Пример моделирования синхронной машины (двигателя)

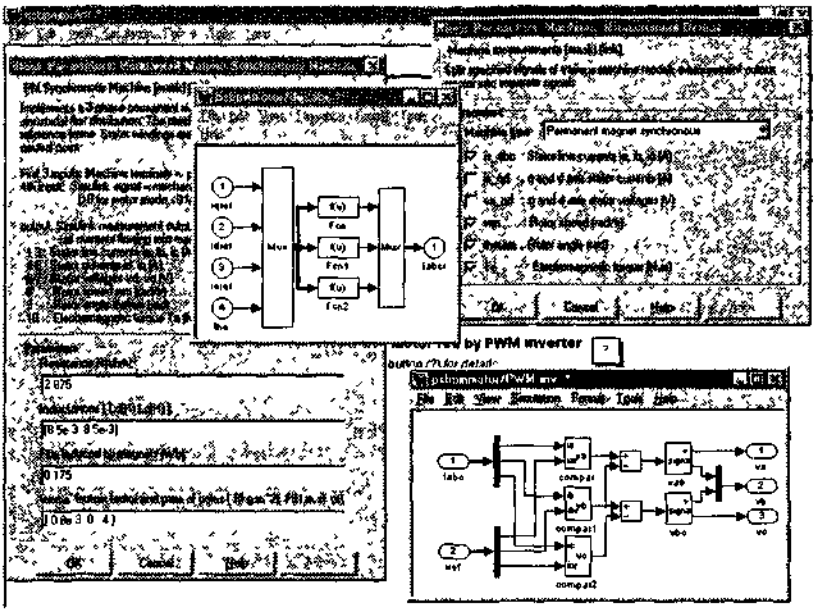


Рис. 15.59. Субсистемы и окно установки параметров для синхронной машины

Моделирование асинхронных машин

Асинхронные машины получили широкое распространение в основном как двигатели для электрического привода. Это связано с их нетребовательностью и простотой. Такие машины получают питание от импульсных преобразователей или от промышленной сети переменного тока.

Пример моделирования асинхронного двигателя, питаемого от трехфазной сети, созданной импульсным преобразователем релейного типа, показан на рис. 15.60. Осциллограммы характеризуют работу системы на этапе разгона двигателя.

На рис. 15.61 показаны окна установки параметров асинхронной машины и окно измерения ее параметров. Последнее окно дает представление о системе параметров асинхронной машины, доступных для регистрации с помощью виртуальных измерительных приборов.

Следует отметить, что асинхронные машины в пакете Power System Blockset представлены двумя моделями: pu Units 1 (с трехфазным входом) и pu Units (со входом, имеющим две фазы с нейтралью).

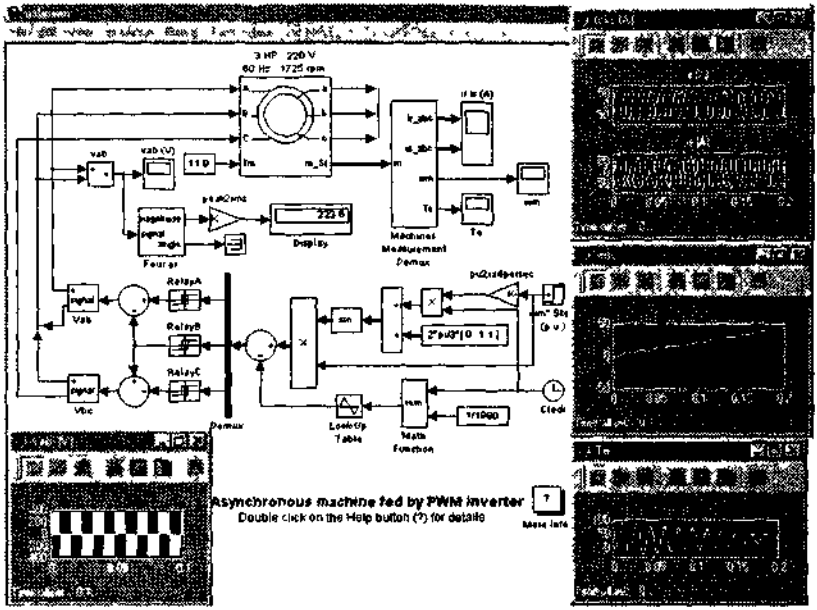


Рис. 15.60. Моделирование привода асинхронного двигателя, питаемого от трехфазного импульсного источника

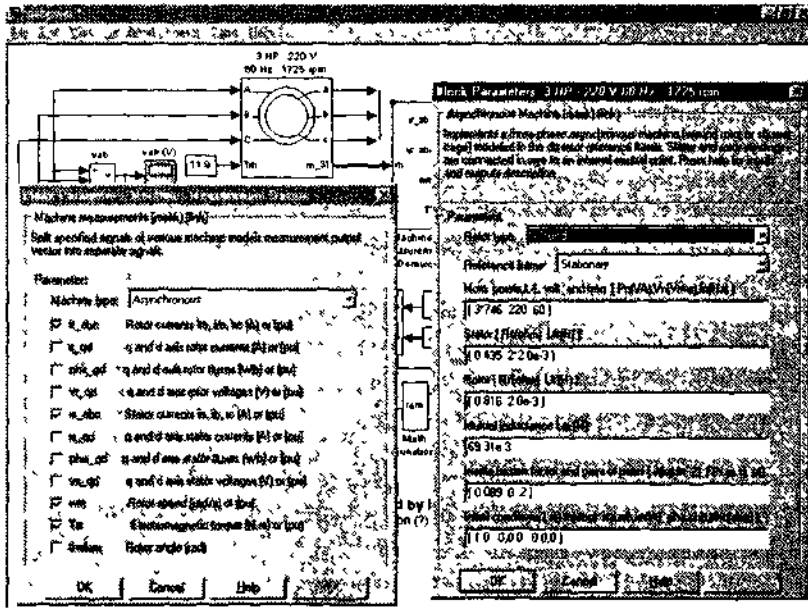


Рис. 15.61. Окна установки параметров асинхронной машины и окно контроля измеряемых параметров

Дополнительные возможности пакета Power System

Моделирование линии передачи электроэнергии с компенсаторами

На рис. 15.62 показан пример моделирования переходных процессов в высоковольтной линии передачи электроэнергии с напряжением 735 кВ, имеющей два участка длиной 150 км каждый и индуктивные компенсаторы.

Осциллограммы, иллюстрирующие работу линии с компенсаторами, представлены на рис. 16.63. Они позволяют судить о сложности переходных процессов, возникающих при коммутации линий электропередачи.

Применение графического интерфейса пользователя

В одном из примеров (рис. 15.34) мы уже отмечали возможность применения средств графического интерфейса пользователя (GUI) для контроля за процессом моделирования системы или устройства. Эти средства можно использовать как для входящих в библиотеку примеров, так и для составляемых пользователем.

Для этого достаточно ввести блок powergui в окно модели. Этот блок находится в разделе Power System Blockset основной библиотеки Simulink. После этого в выбранной модели становятся доступны все возможности GUI.

Пример применения блока powergui представлен на рис. 15.62. Интерфейс powergui имеет несколько режимов работы:

- States — вывод данных о состояниях модели;
- Measurements — вывод данных измерений;
- Sources — вывод данных об источниках сигналов;
- Nonlinear — вывод данных о нелинейных параметрах.

Меню интерфейса имеет два пункта:

- Tools — доступ к средствам инициализации и контроля состояний;
- Report — генерация отчета о моделировании.

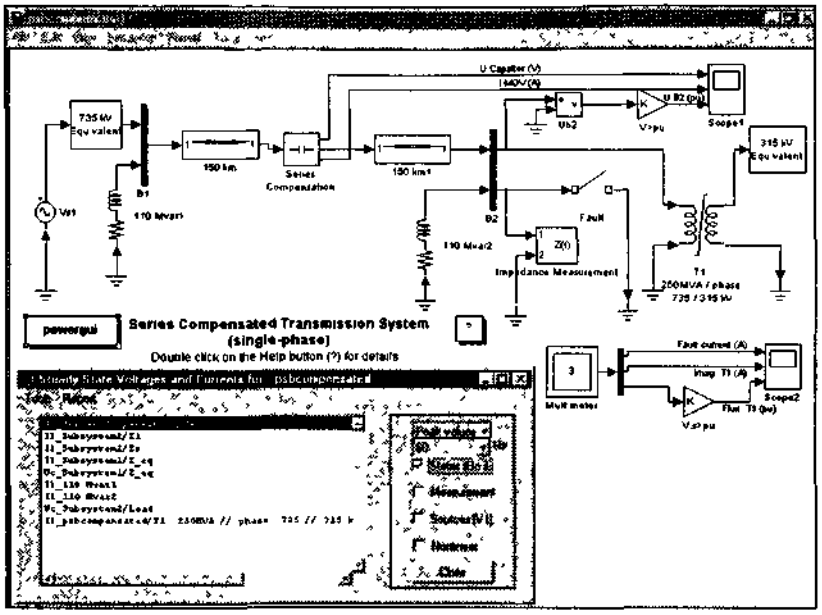


Рис. 15.62. Моделирование линии электропередачи с компенсаторами

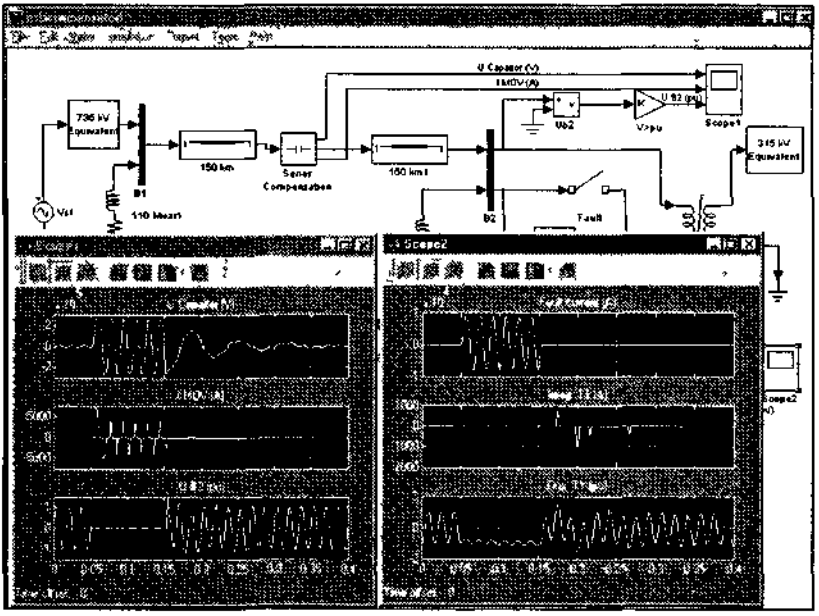


Рис. 15.63. Осциллограммы модели рис. 13 62

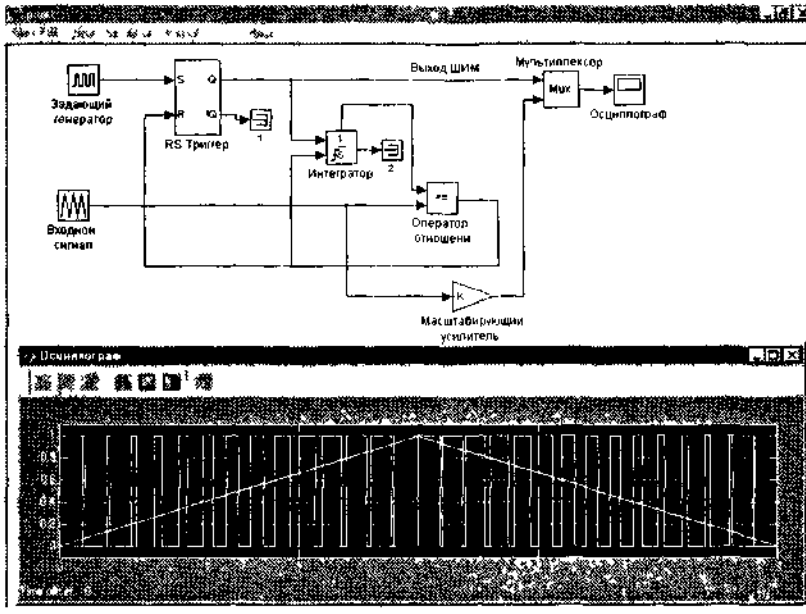


Рис. 15.64. Подсистема ШИМ

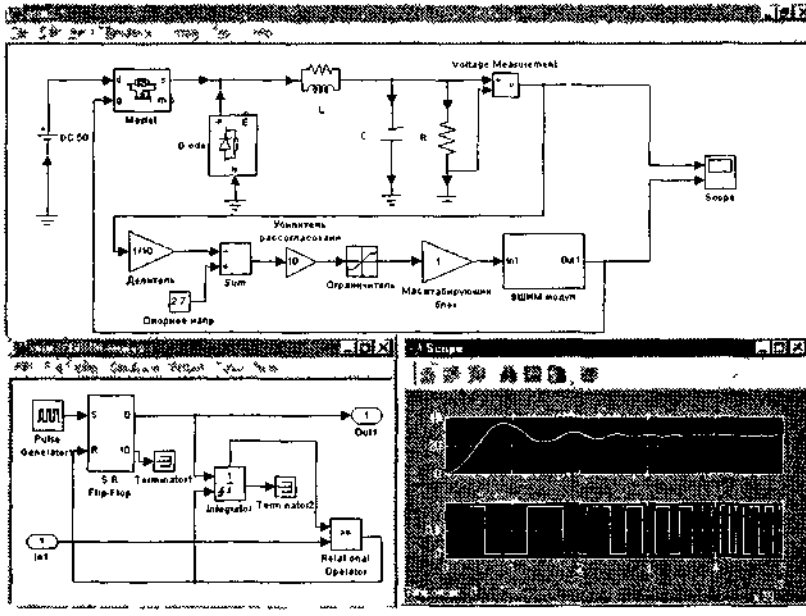


Рис. 15.65. Моделирование преобразователя с идеальным ключом, управляемым ШИМ

Построение собственной субмодели ШИМ

Описанная выше модель преобразователя (рис. 15.40) идеализирована и не содержит элементов автоматического управления. Обычно преобразователи такого рода используют для управления ключом широтно-импульсный модулятор. Рисунок 15.64 показывает одну из реализаций ШИМ, а также управляющий треугольный сигнал на его входе и выходные импульсы модулятора. В отличие от ранее рассмотренных примеров этот пример не является фирменным демонстрационным примером.

Модель построена с применением средств разработки пакета Simulink. Обратите внимание на то, что все пояснения на этой модели даны на русском языке, поскольку модель не является встроенной в библиотеку примеров системы MATLAB.

Моделирование импульсного преобразователя с ключом на полевом транзисторе

На рис. 15.65 показана модель импульсного преобразователя с ключом на мощном полевом транзисторе, 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) на нагрузке 5 Ом, зашунтированной конденсатором 100 мкФ. Если выходное напряжение превышает опорное, скважность импульсов ШИМ уменьшается и ключевой регулятор понижает напряжение на выходе

Осциллограмма выходного напряжения, представленная на рис. 15.65, показывает, что вначале имеет место заметное перерегулирование, дающее короткий всплеск напряжения на выходе примерно до 35 В. Затем напряжение на выходе опускается до уровня примерно 27 В и после некоторого переходного процесса 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 устройства. В частности, наличие перерегулирования и отличие частоты запуска ключа от номинальной (задается источником Pulse Generator и

равна 200 кГц) в начале переходного процесса свидетельствует о неустойчивой работе ШИМ в начале запуска. Однако в конце переходного процесса ШИМ работает устойчиво.

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 ПК с процессорами класса Pentium II и Pentium III. Однако, учитывая дешевизну работы ПК, даже такое время моделирования следует признать вполне приемлемым (известны случаи, когда моделирование сложных моделей занимало многие дни и даже месяцы, оставаясь вполне рентабельным делом).

Функция power2sys

Для детального анализа моделируемых энергетических систем и электрических цепей в MATLAB имеется специальная функция power2sys. Эта функция имеет следующий синтаксис:

```
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 в Simulink под Windows. Эта функция рассматривает линейную часть модели анализируемой схемы и создает ее описание. Оно базируется на следующей матричной системе уравнений:

$$dx/dt = Ax + Bu$$
$$y = Cx + Du$$

Здесь A, B, C и D — матрицы, которые Simulink формирует при составлении матричной модели анализируемой схемы. Смысл этих и других параметров поясним примером.

Запустим схему из файла **psbnetsim2.mdl**. Это можно сделать, указав имя файла прямо в командной строке MATLAB или загрузив файл с помощью пункта меню **File ▸ Open** расширения Simulink. Соответствующая модель появится в окне Simulink (рис. 15.66). Запустив модель на исполнение, получим осциллограмму напряжения на RL-цепи, возникающего при включении ключа.

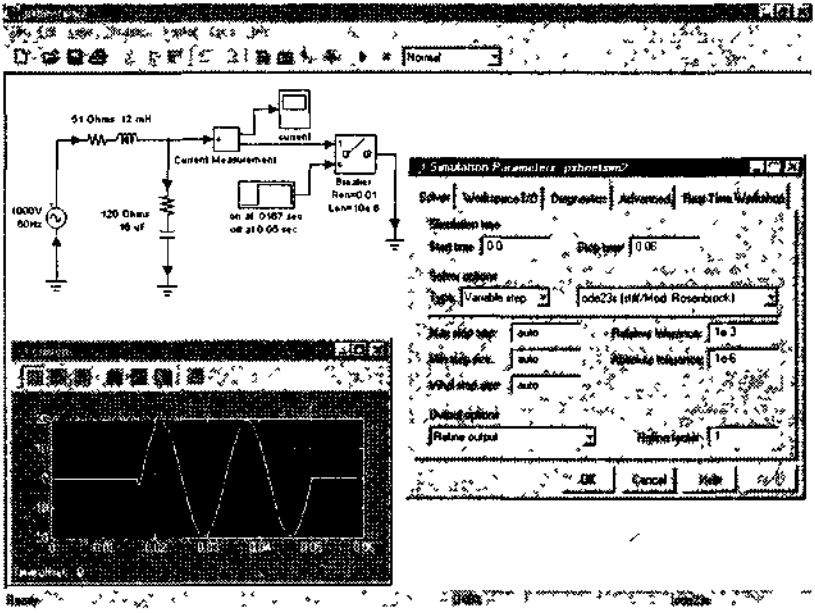


Рис. 15.66. Модель из файла psbnetsim2 mdl

Перейдя в окно MATLAB, выполним команду:

```
>> [A  B  C  D  x0  state_var  inputs  outputs  uss  xss  yss  freqyss
Hlin]= POWER2SYS('psbnetsim2 )
```

Появятся данные о схеме (точнее говоря, о ее линейной части) в следующем виде:

```
Computing state-space representation of linear electrical circuit (V2 1)
(2 states   2 inputs   3 outputs)
A =
1 0e+004 *
        0                6 2500
        -0 0083    -1 4250
B =
  1 0e+004 *
        -6 2500    0
        1 0000           0 0083
C =
        1          120
        0          0
        0          0
D =
  -120       0
        1          0
        1          0
```

```
x0 =
      -513 1443
      2 9190
state_var =
Uc_120 Ohms 16 uF
Il_51 Ohms  12 mH
inputs =
I_Breaker  Ron=0 01 Lon=10e 6
U_1000V 60Hz
outputs =
U_Breaker  Ron=0 01 Lon=10e 6
I_Breaker  Ron=0 01 Lon=10e 6
I_Current Measurement
uss =
                    0
                 1000
xss =
  1 0e+002 *
      4 8392   5 1314i
      0 0310 + 0 0292i
yss =
      1 0e+002 *
      8 5535   1 6287i
               0
               0
freqyss =
      60
Hinn =
  -44 3596 + 4 4368i  0 8553 - 0 1629i
  1 0000                    0
  1 0000                    0
```

Эти данные вряд ли представят интерес для большинства пользователей, которых интересует 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 проявления процесса моделирования. Помимо указанной функции в пакет Power System Blockset входит функция circ2ss, предназначенная для построения статической модели анализируемой системы. Эта функция также вызывается из командной строки MATLAB. Ввиду ограниченного применения детально она не рассматривается, подробное описание этой функции можно найти в справочной системе пакета Power System Blockset.

Частотный анализ цепей

Мы уже отмечали возможность анализа цепей в частотной области. Показанная на рис. 15.67 модель дает еще один пример частотного анализа электрической цепи. Она реализуется с помощью блока powergui

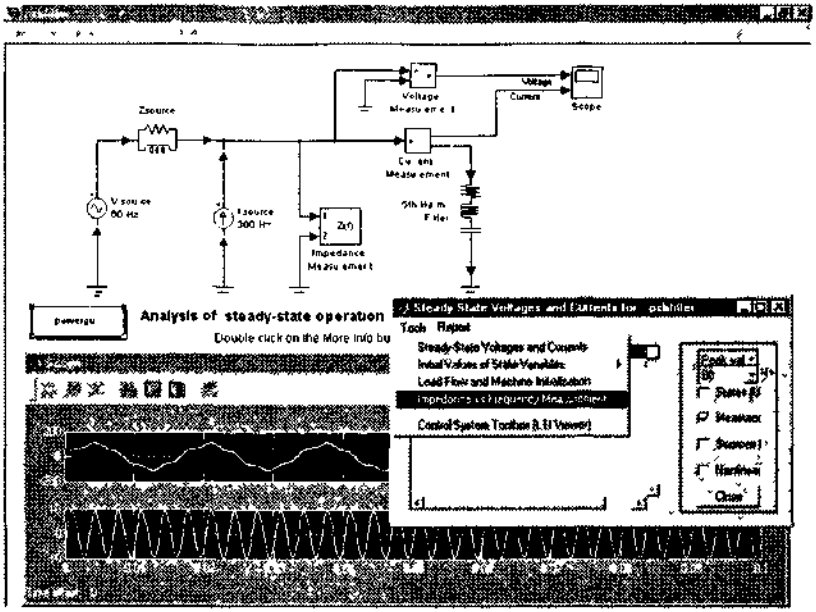


Рис. 15.67. Модель цепи и временные диаграммы ее работы

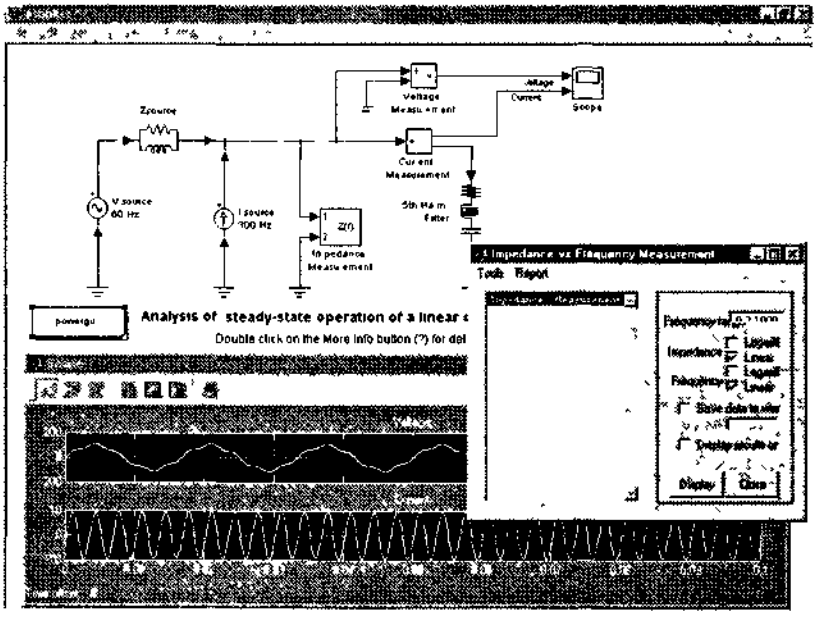


Рис. 15.68. Подготовка к частотному анализу цепи, представленной на рис. 15.67

Нажав на пиктограмму блока powergui, можно вывести окно этого блока (рис. 15 67). Далее следует выбрать пункт меню Tools ▸ Impedance vs Frequency Measurement. В результате данное окно сменится окном Impedance vs Frequency Measurement (см. рис. 15.68).

Теперь остается нажать в этом окне кнопку Display. Появится окно с изображением частотной зависимости полного импеданса цепи от частоты и зависимости фазового сдвига от частоты (рис. 15.69).

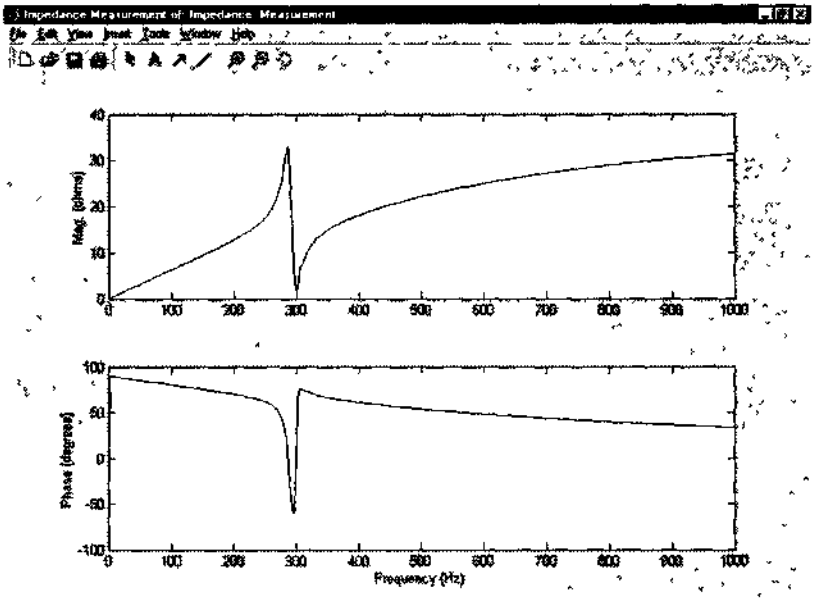


Рис. 15.69. Частотные зависимости импеданса и фазы для цепи модели, показанной на рис 15 67

Библиотеки Extra Library

Внимательный читатель, возможно, заметил, что выше были приведены примеры моделирования электрических машин постоянного тока без ссылок на библиотеки (за исключением модели машины со встроенными магнитами). Эти модели, как и многие другие дополнительные модели, входят в особую библиотеку — Extra Library. Набор компонентов этой библиотеки представлен на рис. 15.70 в окне браузера библиотек.

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.

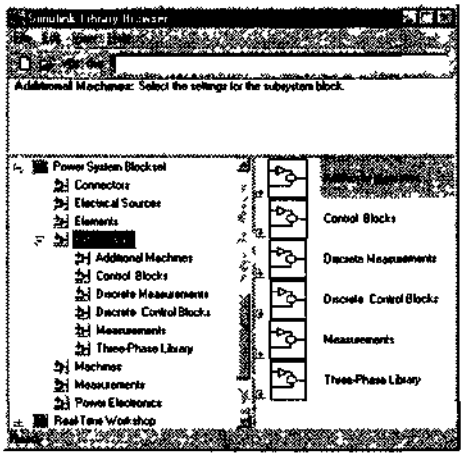


Рис. 15.70. Набор компонентов библиотеки Extra Library

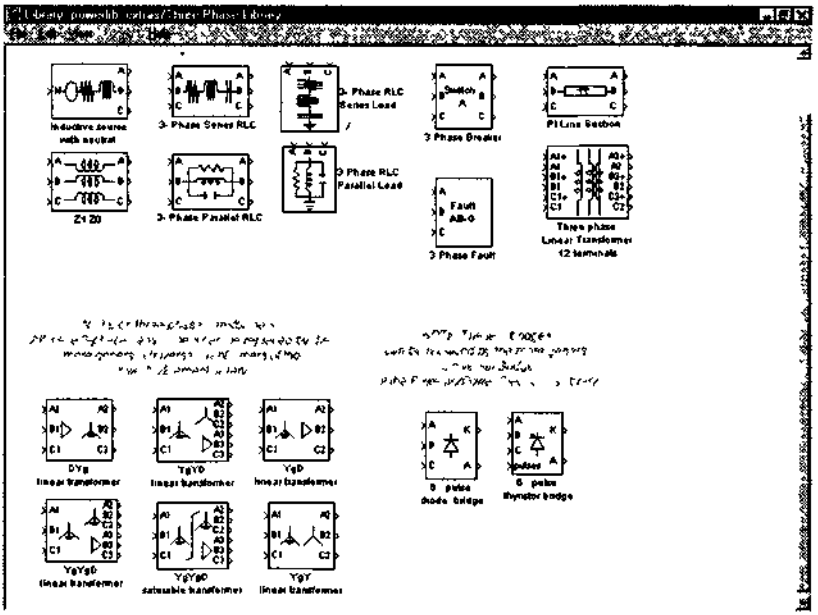


Рис. 15.71. Раздел трехфазных компонентов библиотеки Extra Library

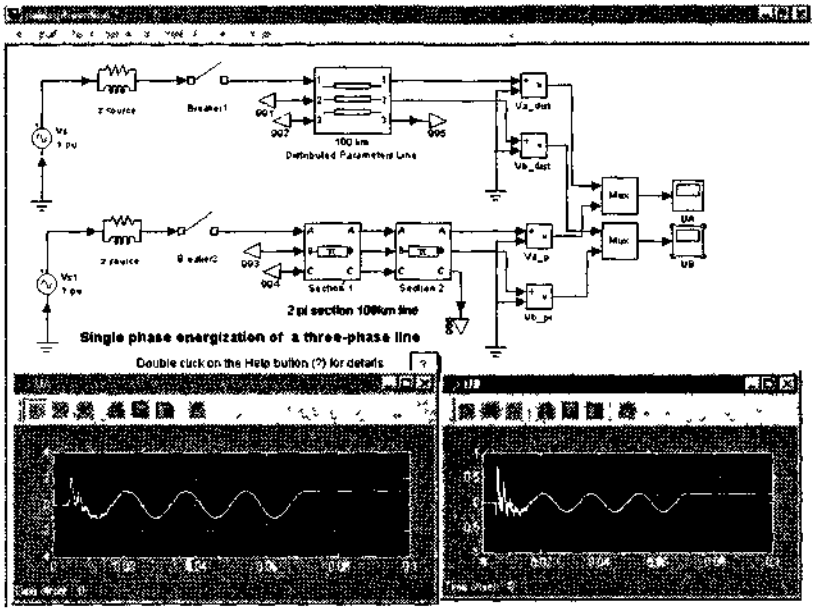


Рис. 15.72. Сравнение переходных процессов в трехфазных линиях разного типа

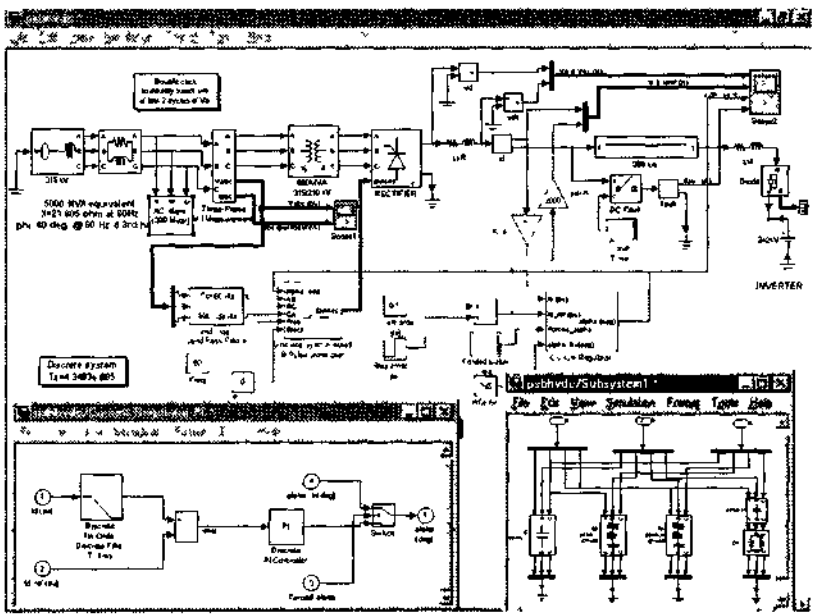


Рис. 15.73. Моделирование сложной энергосистемы большой мощности

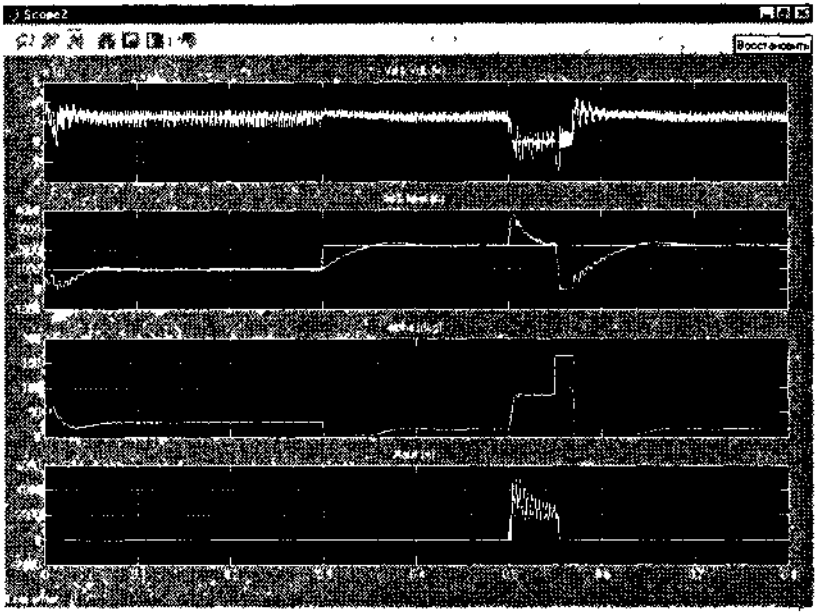


Рис. 15.74. Результаты моделирования энергосистемы большой мощности рис 15 73

Моделирование трехфазных линий передачи

Наиболее обширным является раздел библиотеки Extra Library, посвященный трехфазным системам и устройствам (рис 15 71). Это и понятно — такие устройства получили в промышленности очень широкое распространение

В качестве примера применения библиотеки Extra Library рассмотрим моделирование переходных процессов в трехфазных линиях с сосредоточенными и распределенными параметрами (рис. 15 72).

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 распределенными параметрами.

Моделирование сложной энергетической системы

В заключение рассмотрим моделирование сложной энергетической системы, показанной на рис. 15.73. Эта система содержит мощный трехфазный источник (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 переходных процессах в этой системе.

В этой системе имеется ряд подсистем Две из них представлены на рис. 15.73: регулятор тока PI и мощный трехфазный реактивный фильтр. В фильтре использован целый ряд трехфазных компонентов из библиотеки Extra Library. Осциллограммы работы системы, полученные виртуальным четырехканальным осциллографом, приведены на рис. 15.74.

Таким образом, система рис. 15.73 содержит множество разнохарактерных и взаимодействующих друг с другом устройств. В этих условиях выбор метода решения дифференциальных уравнений представляет заметные трудности. В данном случае разработчики модели отказались от применения методов с переменным шагом интегрирования в пользу метода с постоянным шагом. Следовательно, по существу, данная модель является дискретной моделью энергосистемы.

Глава 16. Пакеты Stateflow и Real-Time Workshop

- Назначение пакета событийного моделирования Stateflow
- Основные объекты SF-диаграмм
- Создание SF-диаграмм
- Запуск и отладка SF-диаграмм
- Оформление SF-диаграмм
- Демонстрационные примеры пакета Stateflow
- Мастерская реального времени Real-Time Workshop

Обзор пакета событийного моделирования Stateflow

Назначение пакета Stateflow 4.0

Пакет событийного моделирования Stateflow (новая версия 4.0) основан на теории конечных автоматов. Он позволяет представить функционирование системы на основе цепочки правил, которые задают соответствие событий и действий, выполняемых в ответ на эти события. 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 Simulink и ряда пакетов расширения (Control System, Digital Signal Processor и др.);
- наглядные интерактивные уроки по моделированию и проектированию систем.

Взаимодействие пакета Stateflow с другими компонентами системы MATLAB представлено на диаграмме рис. 16.1. Из нее видно, что пакет занимает важное место в иерархии пакетов расширения системы MATLAB.

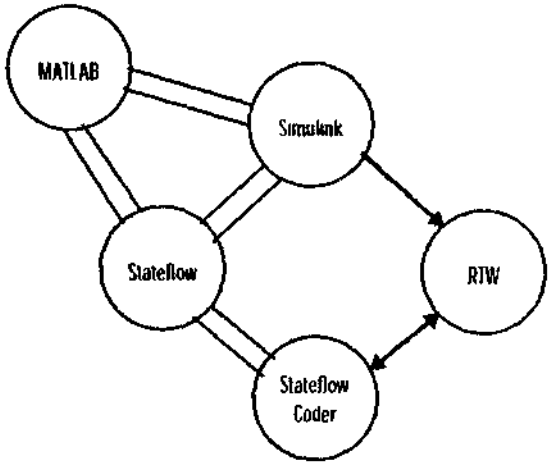


Рис. 16.1. Место пакета Stateflow в системе пакетов расширения MATLAB

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 отражаются все изменения модели: например, строятся диаграммы переходов, изменяются цвета блоков в зависимости от их активности и т. д. Это позволяет визуально отслеживать поведение системы в ходе моделирования. К сожалению, в примерах книги трудно в полной мере отразить динамику SF-диаграмм.

Графический интерфейс пользователя пакета Stateflow имеет следующие компоненты.

- графический редактор SF-диаграмм;
- обозреватель для анализа SF-диаграмм (Stateflow Explorer);
- навигатор (Stateflow Finder) для поиска в SF-диаграммах нужных объектов;

- отладчик SF-моделей;
- генератор кодов для работы совместно с расширением Real Time Workshop.

Эти компоненты обеспечивают интуитивно понятные и простые приемы работы с пакетом Stateflow.

Доступ к средствам Stateflow

Для обеспечения доступа к средствам пакета расширения Stateflow пакет нужно установить. Это делается в ходе инсталляции системы MATLAB отметкой приложения Stateflow Coder в окне компонентов системы MATLAB. После этого в окне браузера библиотек, вызванного запуском системы Simulink, появится раздел Stateflow.

На первый взгляд средства пакета Stateflow очень просты. Они используются совместно с составленной пользователем моделью путем вставки в нее блока библиотеки Stateflow с именем Chart (Диаграмма). Доступ к блоку Chart обеспечивается браузером библиотек. Найдя библиотеку Stateflow, нужно открыть ее окно (рис. 16.2), используя контекстное меню с единственной командой.

В окне библиотеки (основном) присутствует единственный блок Chart. Его можно перетащить мышью в окно модели Simulink.

Доступ к демонстрационным примерам

В новой версии Simulink доступа к демонстрационным примерам из окна основной библиотеки уже нет. Поэтому для открытия окна с демонстрационными примерами надо выбрать пункт меню File ▸ Open и найти файл sf_exmples в папке toolbox\stateflow\sfdemos. Окно с демонстрационными примерами показано на рис. 16.2 в правом нижнем углу окна модели Simulink.

Подключение блока Chart к модели Simulink

Средства Stateflow могут использоваться как самостоятельно (см. ниже), так и в составе моделей Simulink. Все, что надо сделать для подключения блока Chart к модели в Simulink, — это перетащить его мышью в окно модели и подключить к нужному месту. Рисунок 16.3 показывает заготовку простой модели Simulink, состоящей из источника сигнала с осциллографом. В нее внедрен блок Chart пакета Stateflow.

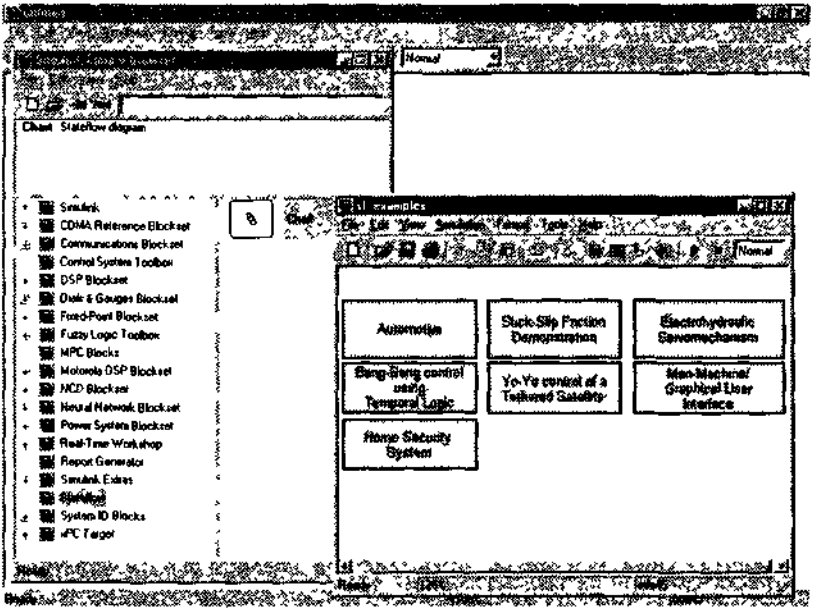


Рис. 16.2. Окна библиотеки и примеров применения системы Stateflow

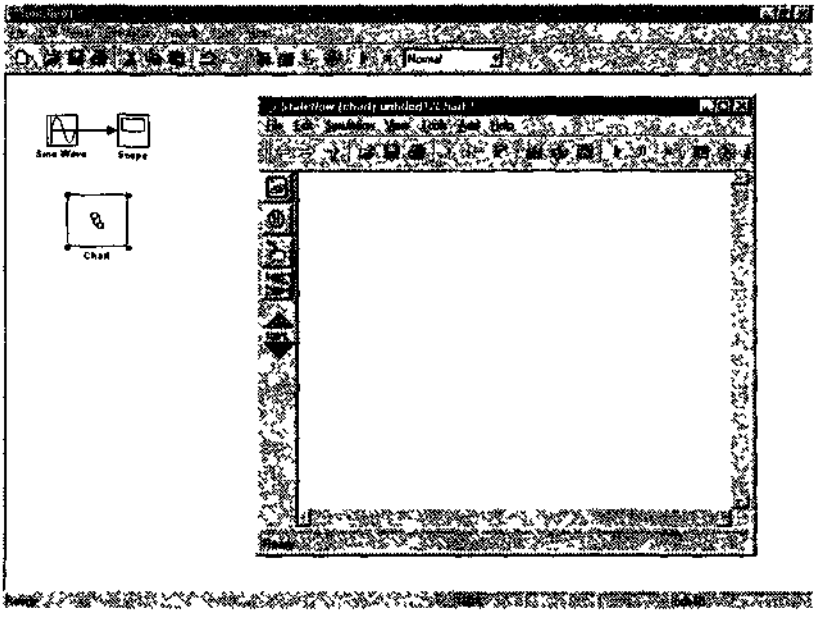


Рис. 16.3. Простая модель Simulink с блоком SF-диаграммы

Первоначально активизация этого блока вызывает пустое окно редактора SF-диаграмм, также показанное на рис. 16.3. Его можно использовать для создания SF-диаграмм и их отладки с целью получения от них нужных функций. Совокупность всех SF-блоков, имеющихся в составе модели, образует *Stateflow-машину*. Как превратить заготовку рис. 16.3 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 графическая диаграмма, создаваемая средствами графического интерфейса пакета расширения Stateflow. SF-диаграмма служит для визуального представления работы моделируемой системы. Это достигается анализом всех стадий ее работы с указанием активных и пассивных в данное время блоков и 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 материалах книг с черно-белыми рисунками.

Прежде чем описывать средства создания конкретных SF-диаграмм, рассмотрим типичную SF-диаграмму некой абстрактной системы. Такая диаграмма с обозначенными на ней объектами представлена на рис. 16.4.

На рис. 16.4 показаны основные типы объектов SF-диаграмм. Позже мы рассмотрим их более подробно.

Работа с редактором SF-диаграмм

Только что созданный блок Chart представляет пустую диаграмму — заготовку для создания действующей диаграммы. Если активизировать мышью пустой блок Chart, то появится окно редактора SF-диаграмм, в котором можно начать построение диаграммы (рис. 16.5). Перетаскивая мышью пиктограммы панели инструментов, находящейся в правой части окна редактора, можно поместить в SF-диаграмму нужные компоненты.

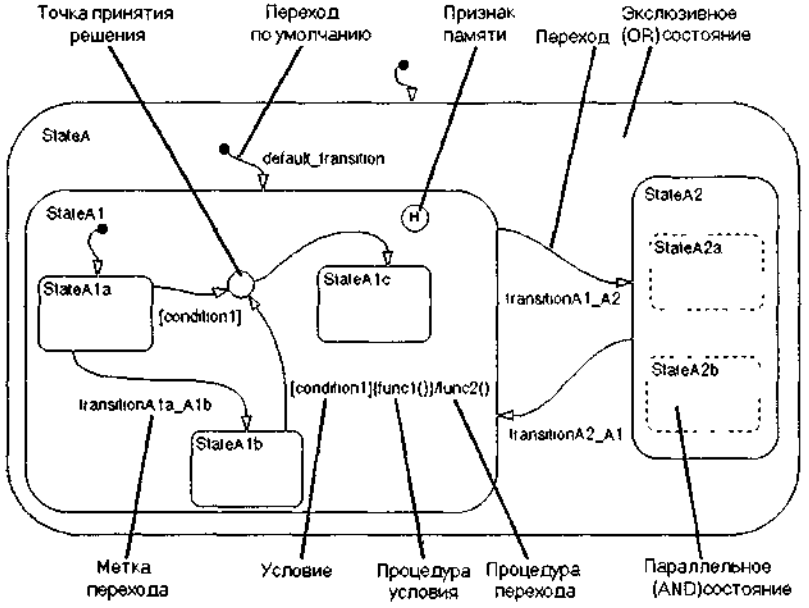


Рис. 16.4. Вид Stateflow-диаграммы

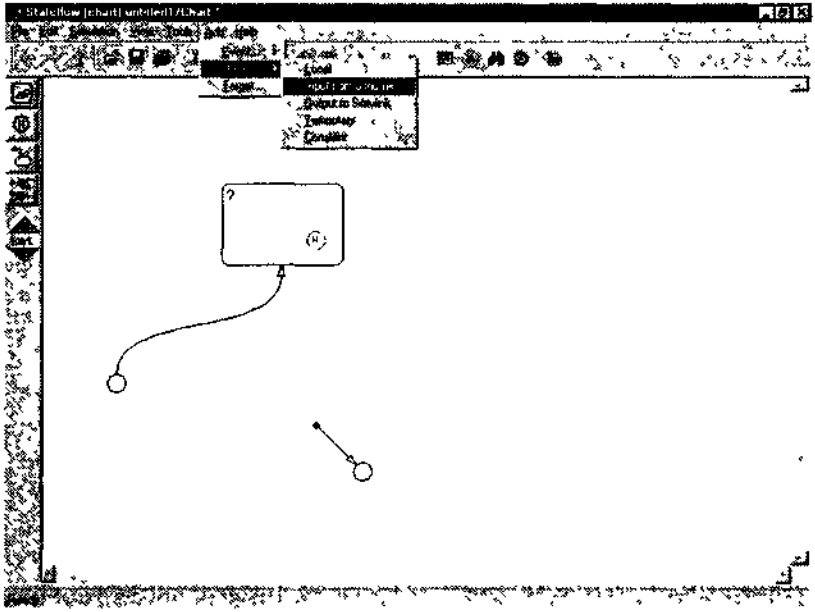


Рис. 16.5. Окно Stateflow-диаграммы с ее основными компонентами

Редактор SF-диаграмм имеет традиционный для окон системы MATLAB вид. Основные средства построения диаграмм сосредоточены в меню Add (Добавить). Эти средства дублируются в панели инструментов, которая располагается у правой границы окна. Ее четыре кнопки имеют следующее назначение (сверху вниз):

- установка состояния State (или вложенных состояний);
- установка признака «состояние с памятью» History junction;
- установка перехода по умолчанию Default transition;
- установка признака альтернативы Connective junction.

Названия кнопок (Name), изображения соответствующих им блоков (Notation) и вид пиктограмм панели инструментов (Toolbar Icon) представлены на рис. 16.6.

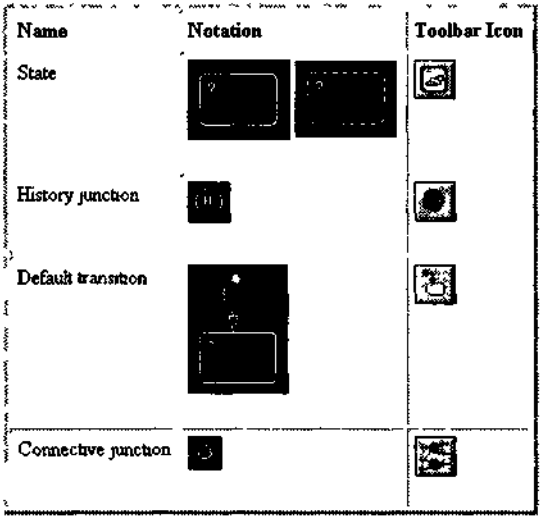


Рис. 16.6. Средства построения SF-диаграммы

Блоки можно переносить мышью в окно SF-диаграммы, а затем соединять их друг с другом с помощью мыши при нажатой левой кнопке. При этом строятся в общем случае изгибающиеся стрелки, причем изгиб можно формировать также перемещением мыши. На рис. 16.5 показана SF-диаграмма с четырьмя основными графическими объектами.

Основные объекты SF-диаграмм

Состояния

Важным объектом SF-диаграммы является *состояние* (state). Это графический объект в виде прямоугольника со скругленными углами, построенный обычно синими линиями. Свойства активности и пассивности состояния динамически сменяют друг друга в зависимости от происходящих событий. Каждое состояние имеет своего родителя и может иметь потомков (состояния более низкого уровня). Если состояние является единственным, то его родителем является сама SF-диаграмма, называемая также *корневой диаграммой*.

Диаграмма состоит из вложенных друг в друга состояний. Основное состояние называется эксклюзивным. Если строится единственное состояние, то оно и будет эксклюзивным. Каждое состояние может быть родителем, то есть иметь своих потомков. Если в какое-либо состояние включить другое состояние, то первое станет родителем.

Состояния могут быть нейтральными (neutral) и занятыми (engaged). Имеются два типа состояний: параллельные, существующие одновременно (AND, или И), и взаимно исключающие друг друга (OR, или ИЛИ). Например, взаимно исключающими друг друга являются состояния двустабильного выключателя «Включено» и «Выключено».

Признаки памяти

Каждое состояние имеет *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 высший приоритет выполнения. Она изображается как красная окружность с буквой H внутри.

Переходы

Переходы (transition) — еще один важный графический объект SF-диаграмм. Переходы отражают связь одного объекта с другим и представляются обычно красными стрелками. Переходы не имеют своей кнопки в панели инструментов. Они создаются, когда вы, указав курсором мыши (при нажатой левой кнопке) объект, от которого начинается стрелка перехода, двигаете курсор к другому объекту. Переходы имеют *метки*, описывающие обстоятельства или условия,

при которых происходит переход от одного состояния к другому. 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 черной окружности, закрашенной внутри красным цветом, имеющие стрелку перехода. Применение таких объектов исключает пересечения переходов и упрощает построение SF-диаграмм. Кроме того, альтернативные пути переходов способствуют повышению эффективности работы SF-диаграмм и облегчают генерацию программного кода.

События

Событие (event) — важнейшее понятие пакета Stateflow. Возникновение события меняет статус связанных с ним состояний и может запустить действие или переход, связанные с ним. При этом события распространяются сверху вниз (от события-родителя к событию-потомку).

Событие не является графическим объектом. Однако для визуализации события можно использовать метки переходов, связанные с ним. Каждое событие должно быть определено с помощью того или иного *условия*, записанного как логическое выражение.

Рисунок 16.7 иллюстрирует построение двух состояний: нейтрального (neutral) и занятого (engaged).

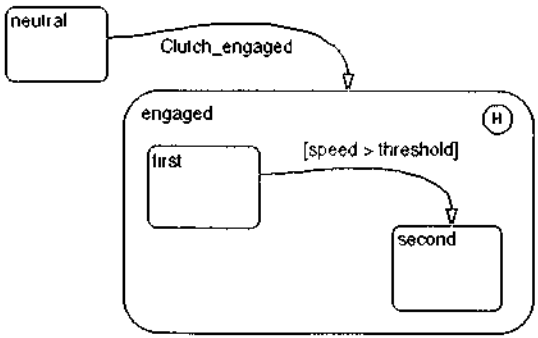


Рис. 16.7. Пример простой SF-диаграммы

Затем между ними создается переход clutch_engaged. После этого создаются два состояния-потомка: first и second. Между ними организован переход по условию (если скорость speed превышает заданную величину threshold). Обратите внимание, что условие записывается в квадратных скобках. Наконец, вводится история занятого объекта — в его правом верхнем углу.

События имеют свойства. Главным из них является свойство видимости события. В зависимости от значения свойства видимости события могут быть следующих типов:

- локальные, то есть видимые только в пределах заданной SF-диаграммы;
- входные — передаваемые в SF-диаграмму из модели Simulink;
- выходные — передаваемые из SF-диаграммы в модель Simulink;
- 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, дан на рис. 16.7. Здесь можно выделить процедуру условия и процедуру перехода.

Пример задания процедуры, связанной с состоянием объекта, дан на рис. 16.8.

При записи процедур используется семантика пакета Stateflow. Из-за ограниченного объема книги подробно семантика записи действий не рассматривается. Такого рассмотрения, кстати, нет и в объемном фирменном описании — в нем семантика излагается с привлечением множества практических примеров. Демонстрационные примеры Stateflow неплохо иллюстрируют семантику записи процедур,

и мы призываем читателя, заинтересованного в глубоком знакомстве с этим вопросом, внимательно проанализировать демонстрационные примеры. Часть из них описана ниже.

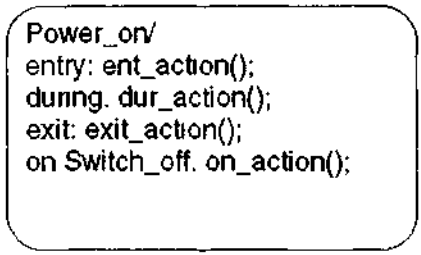


Рис. 16.8. Пример задания процедуры, связанной с состоянием объекта

Данные

Данные (data) в SF-модели являются числовыми значениями. Данные не являются графическими объектами и непосредственно на SF-диаграмме не указываются. Они могут создаваться на любом уровне иерархии модели и имеют свойства. Данные, имеющие свойство видимости, могут быть следующих типов:

- локальные, то есть видимые только в пределах заданной SF-диаграммы;
- входные — передаваемые в SF-диаграмму из Simulink-модели;
- выходные — передаваемые из SF-диаграммы в Simulink-модель,
- временные (или промежуточные);
- сохраняемые (в рабочем пространстве MATLAB);
- константы;
- экспортируемые — передаваемые из SF- или Simulink-модели во внешнюю программу;
- импортируемые — получаемые из внешних программ.

Для создания и модификации данных следует воспользоваться пунктом меню Add ▸ Data редактора SF-диаграмм обозревателем Stateflow.

Описание объектов

При создании состояния в левом верхнем углу его графического изображения появляется знак вопроса. На его месте вводится опи-

сание 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 — процедура, которая выполняется как часть активной процедуры;
- entry: или en — процедура, которая выполняется как часть входной процедуры;
  - entry(имя_состояния) или en(имя_состояния) — генерация локального события при вводе имени состояния;
- exit или ex — процедура, которая выполняется как часть процедуры выхода из состояния;
  - exit(имя_состояния) или ex(имя_состояния) — генерация локального события при удалении указанного состояния;
- on имя_состояния или none — процедура, которая выполняется, когда указанное имя состояния определено с ключевым словом broadcast (это означает, что связь между блоками организована без явного задания линии между ними).

Пример описания состояния дан на рис. 16.7. Аналогичным образом описания других объектов также вводятся по запросу в виде вопросительного знака.

Язык описания процедур Action Language построен на основе синтаксиса языка C и содержит арифметические и логические операторы и функции, некоторые специальные функции и функции пользователя. Отметим следующие функции:

- change(имя_данных) или chg(имя_данных) — генерация локального события в случае, когда меняется значение указанных в аргументе данных;
- in(имя_состояния) — функция условия, дающая при оценке значение true в случае, когда указанное в качестве аргумента состояние является активным;
- send(имя_события,имя_состояния) — посылает спецификацию события к спецификации состояния (прямая передача события);

- `matlab(evalString,arg1,arg2,...)` или `ml()` — процедура, выполняющая вычисления, записанные в строке `evalString` в нотации функций системы MATLAB (ml-нотация), с перечисленными вслед за ней аргументами;
- `matlab MATLAB_workspace_data` или `ml` — процедура, выполняющая вычисления с использованием ml-нотации.

Примеры описаний функций проще всего взять из демонстрационных примеров. Особо надо отметить применение символа `t` для обозначения времени, например, в выражениях вида

```
[t - On_time > Duration]
```

Построение SF-диаграмм

Создание модели Simulink с заготовкой SF-диаграммы

Попробуем составить и запустить простую модель электрического выключателя, который в исходном состоянии выключен. Спустя некоторое время, мы включаем его, и напряжение поступает на нагрузку, подключенную через выключатель к источнику переменного напряжения. Для моделирования такого устройства пригодна заготовка модели, показанная на рис. 16.3. Покажем, как на ее основе создается работающая модель с SF-диаграммой.

Прежде всего надо запустить расширение Simulink, например нажав кнопку запуска на панели инструментов MATLAB. Как обычно, появится окно браузера 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 показано на рис. 16.9 в левом верхнем углу окна Simulink-модели.

Далее откроем библиотеку Stateflow и перенесем в окно Simulink-модели блок SF-диаграммы, представленный на рис. 16.9, в слегка растянутом виде. Установив курсор мыши в конец надписи блока Chart и щелкнув левой кнопкой мыши, перейдем в режим редактирования надписи. Заменим эту надпись нужной нам надписью — On-off.

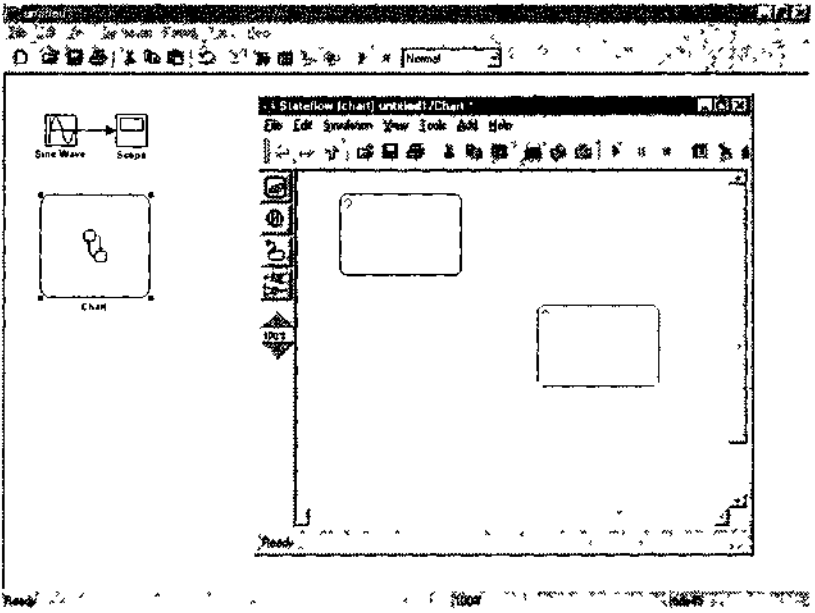


Рис. 16.9. Построение модели выключателя с SF-диаграммой

Начало работы с SF-диаграммой в редакторе SF-диаграмм

Теперь двойным щелчком мыши на SF-блоке On-Off можно открыть окно редактора SF-диаграмм и начать построение SF-диаграммы. Используя первую сверху кнопку панели инструментов редактора SF-диаграмм, перенесем в окно модели два графических объекта типа состояния. Их можно видеть в окне редактора SF-диаграмм, показанного на рис. 16 9 справа.

Пока оба состояния не имеют имен — вместо них стоят вопросительные знаки. Щелкнув на знаке вопроса мышью, можно получить поле с маркером ввода (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 не полный), показанный на рис. 16.10.

Если вас не устраивает местоположение созданных состояний или их размеры, вы можете их изменить. Наведя курсор мыши на середину скругленного прямоугольника и затем нажав левую кнопку

мыши, можно перемещать его мышью по полю экрана и устанавливать в любое место. Ухватившись курсором мыши за угол (курсор при этом приобретает вид двухсторонней стрелки), можно менять размеры прямоугольника.

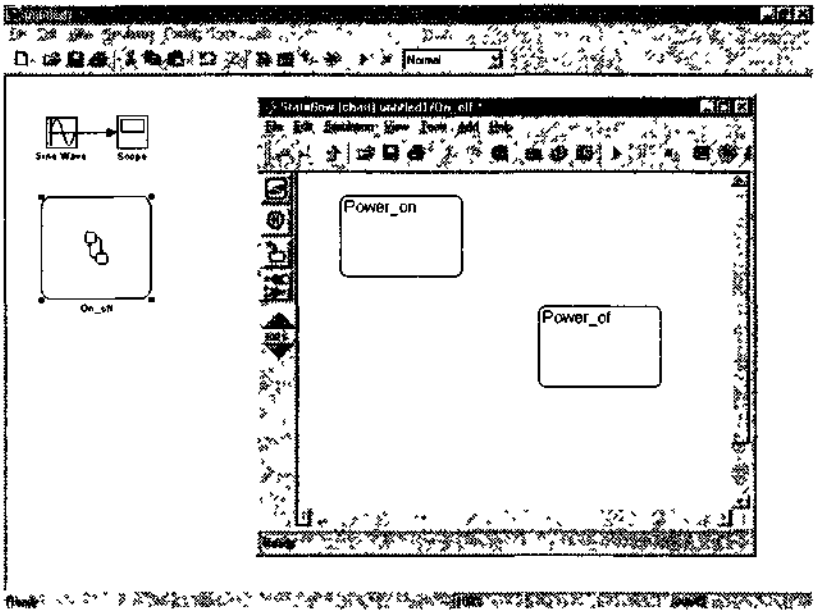


Рис. 16.10. Модель выключателя с SF-диаграммой, содержащей два состояния

Создание переходов между состояниями

Далее можно приступать к созданию переходов между состояниями. Для перехода от состояния Power_on к состоянию Power_off надо установить курсор 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 — их можно выбирать произвольно на задней границе прямоугольника состояния Power_on и передней границе прямоугольника Power_off.

Далее надо построить стрелку перехода от состояния Power_off к состоянию Power_On. После этого SF-диаграмма будет иметь вид, представленный на рис. 16.11 в правом окне.

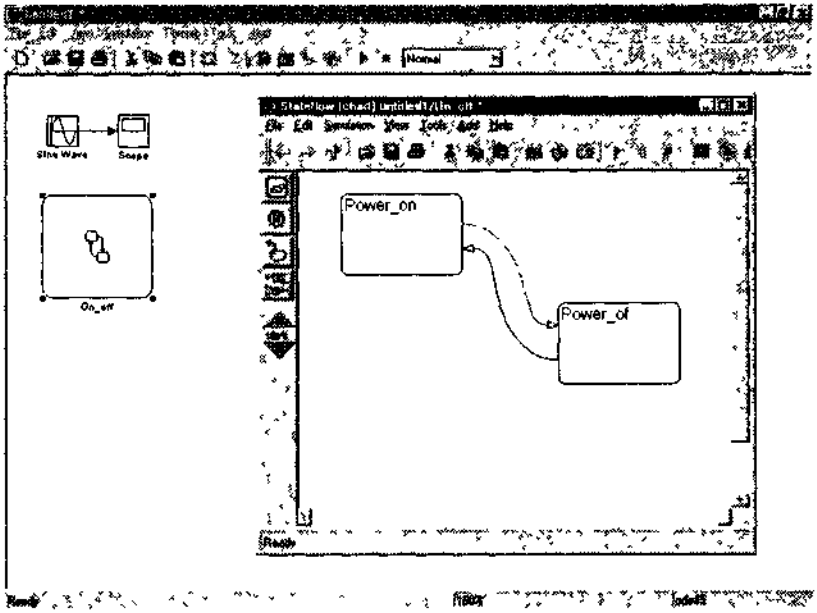


Рис. 16.11. Модель выключателя с SF-диаграммой, дополненной переходами

Обратите внимание, что строящаяся стрелка перехода изгибается при перемещении мыши и этот изгиб можно в дальнейшем корректировать, ухватив стрелку курсором мыши и перемещая ее при нажатой левой кнопке.

Установка названий переходов

Если навести на линию любого перехода курсор мыши и нажать левую кнопку мыши, то появится вопросительный знак, на место которого можно ввести название перехода. Пусть это будет Switch (команда переключения). Но в порядке эксперимента зададим название с ошибкой — Swith.

Установка альтернативного перехода

Кроме того, нужно указать переход Simulink-модели в состояние 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. 16.12.

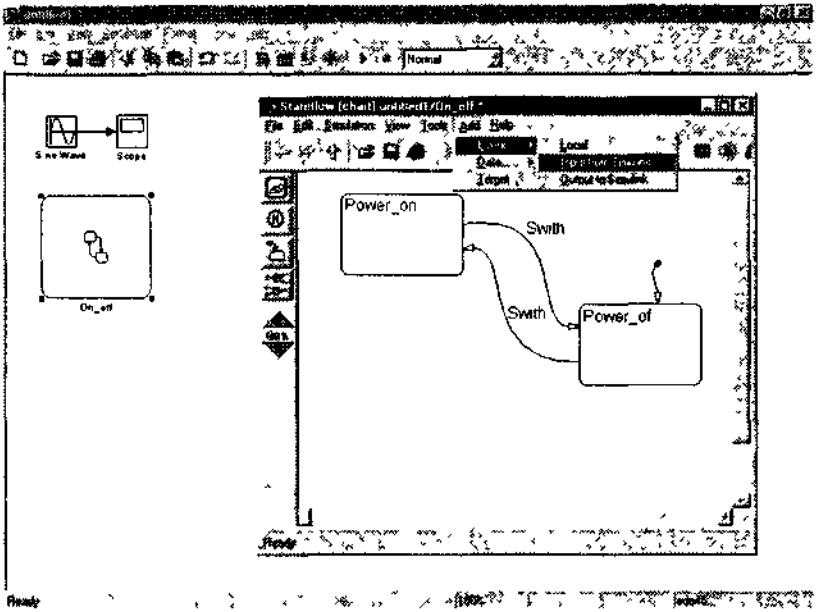


Рис. 16.12. Модель выключателя с законченной SF-диаграммой

На этом графическая часть подготовки модели выключателя заканчивается. Параметры состояний SF-диаграммы можно установить с помощью команды меню Add ▸ Event редактора SF-диаграмм. Эта команда открывает окно с необходимыми установками (рис. 16.13).

Пока можно согласиться с параметрами, заданными по умолчанию. Для этого достаточно нажать клавишу OK.

Если это сделано, то над блоком SF-диаграммы появится триггерный вход, с которым можно соединить выход генератора сигнала. Это будет означать, что мы связали работу генератора сигнала с SF-диаграммой.

Установка параметров SF-диаграммы с помощью обозревателя

Еще один способ установки параметров предоставляет обозреватель Stateflow Explorer (рис. 16.14). Для его запуска используется команда меню Tools ▸ Explore...

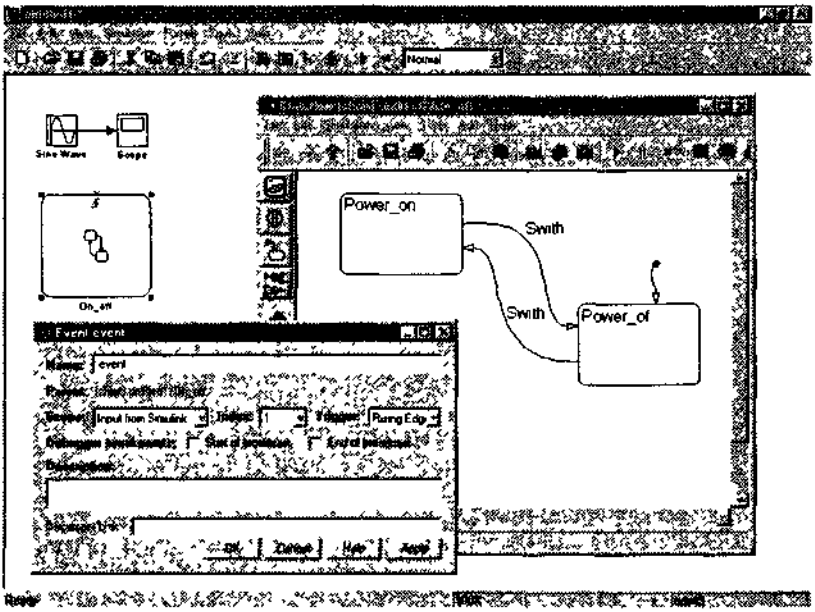


Рис. 16.13. Установка параметров SF-диаграммы с помощью меню редактора SF-диаграмм

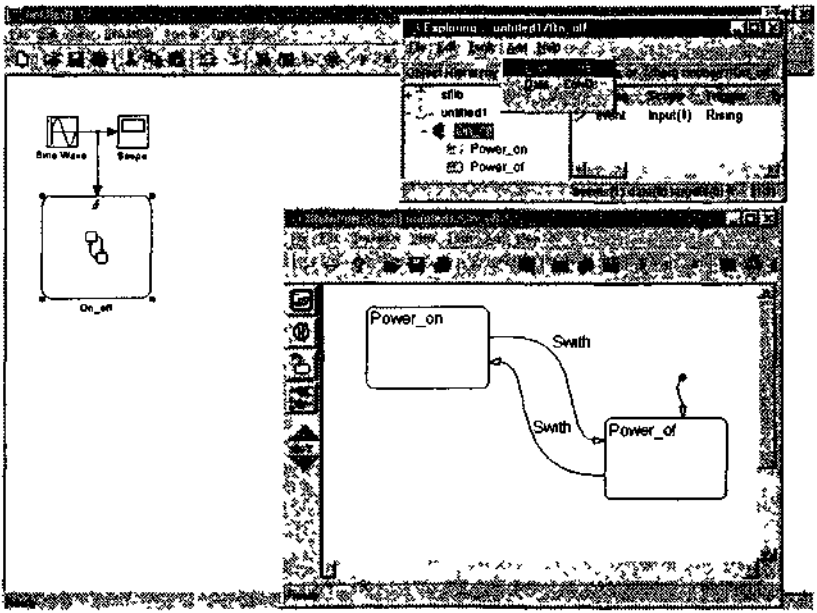


Рис. 16.14. Установка параметров SF-диаграммы в окне обозревателя

В левой половине окна обозревателя (Object Hierarchy) отражена иерархия объектов. В ней можно найти записанную ранее модель sfdemo и входящую в нее SF-диаграмму On_off с объектами Power_on и Power_of. Выделив объект On_off, надо выполнить команду меню обозревателя Add ▸ Event. При этом появится окно Event, с помощью которого можно установить необходимые параметры события. Прежде всего это имя (поле Name), которое надо установить как Switch. В переключателе видимости Scope надо установить значение Input from Simulink, поскольку SF-диаграмма будет работать в Simulink-модели. Наконец, в поле Trigger надо выбрать значение Rising Edge. После этого окно Event и окно обозревателя можно закрыть.

Сохранение модели с SF-диаграммой

Целесообразно сохранить Simulink-модель вместе с SF-диаграммой. Сохраним модель под именем sfdemo. Можно приступить к установке параметров моделирования и пробному запуску.

Запуск и отладка SF-диаграмм

Установка параметров запуска

Перед запуском Simulink-модели следует установить параметры моделирования. Для этого в окне моделей Simulink надо выполнить команду меню Simulation ▸ Parameters... Необходимые установки представлены на рис. 16.15.

На вкладке Solver надо установить начальное Start time и конечное Stop time время моделирования, затем режимы Fixed-step (с фиксированным шагом) и discrete (поскольку работа выключателя носит дискретный характер).

Запуск модели

Запуск производится, как обычно, командой меню Simulation ▸ Start или нажатием кнопки Start в панели инструментов. Можно наблюдать работу модели и SF-диаграммы. Для этого обычно необходимо щелкнуть мышью на блоке SF-диаграммы, чтобы она появилась на переднем плане окна моделей Simulink. Можно запускать ее и отдельно, что описано ниже.

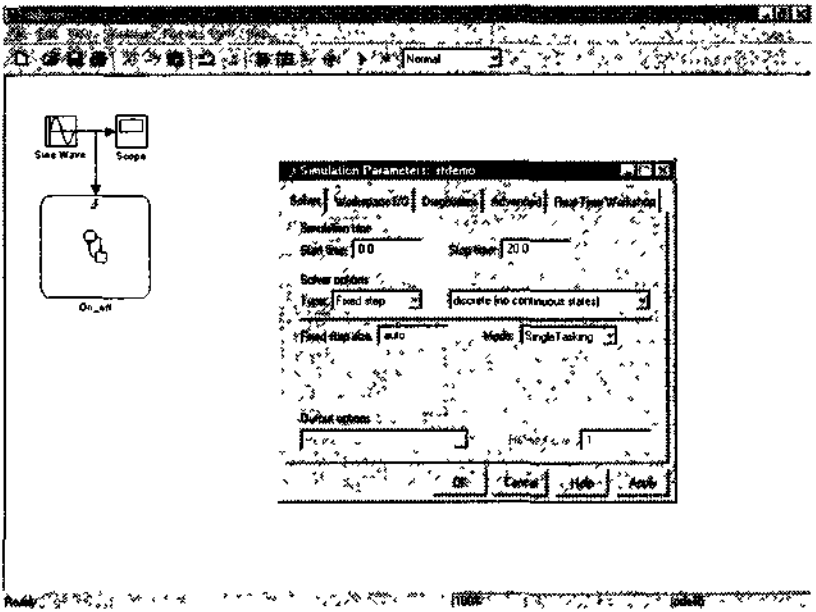


Рис. 16.15. Установка параметров запуска

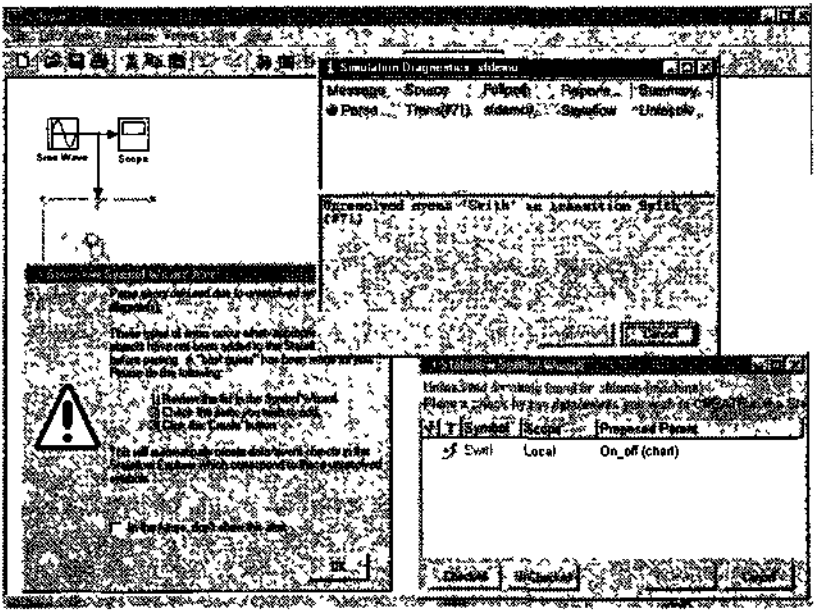


Рис. 16.16. Сообщения об ошибке после запуска модели

Однако пока вместо всего этого мы увидим множество окон с указанием на обнаруженную в диаграмме ошибку — неверную запись команды Switch. Эту ситуацию характеризует рис. 16.16.

Для коррекции ошибок используется редактор SF-диаграмм. Закройте окна ошибок и, открыв редактор SF-диаграмм, исправьте неверные надписи у переходов на правильные — Switch. Заодно исправим и ошибку в наименовании блока Power_off (отсутствие последней f). Проверьте и при необходимости скорректируйте установки параметров. Теперь модель будет иметь вид, представленный на рис. 16.17.

Теперь наша модель должна запускаться как из окна модели Simulink, так и из окна редактора SF-диаграмм.

Работа с отладчиком SF-диаграмм

Поскольку данная модель проста, то моделирование происходит быстро и трудно уследить за его ходом. Поэтому целесообразно воспользоваться специальным отладчиком SF-диаграмм. Его окно (рис. 16.18) появляется при выполнении в окне SF-диаграммы команды меню Tools ▸ Debug....

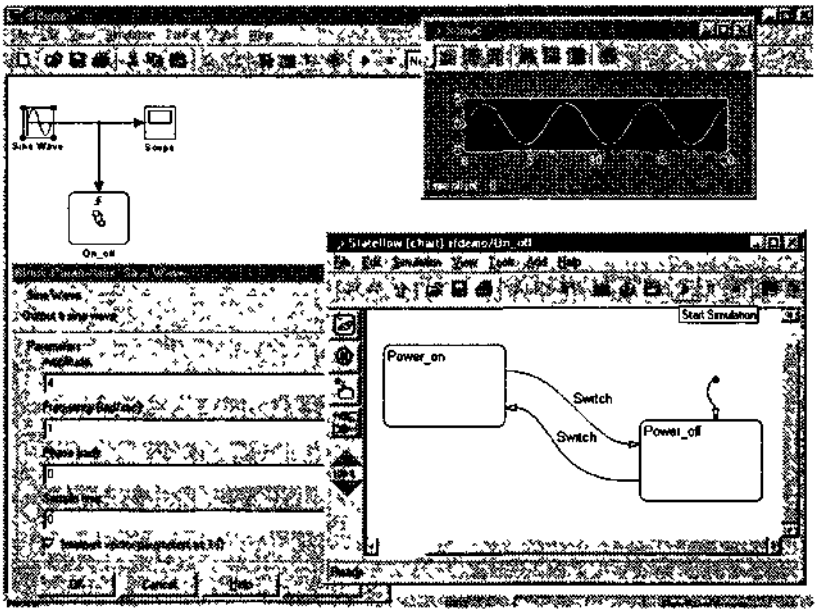


Рис. 16.17. Окончательный вид модели выключателя с SF-диаграммой

Рисунок 16.18 иллюстрирует начало моделирования при нажатии кнопки Start отладчика (она после этого меняет название на Continue). Нетрудно заметить, что сначала активизируется альтернативный переход и состояние Power_off. Они отображаются жирными красными линиями. Активное состояние Power_off означает, что выключатель отключает нагрузку (в нашем случае осциллограф) от источника переменного тока. Поэтому осциллограммы на экране осциллографа нет.

Рисунок 16 19 показывает некоторый промежуточный кадр динамики SF-диаграммы. Видно, что теперь активным становится переход из состояния Power_off в состояние Power_on, что означает активизацию процесса включения нагрузки. На экране осциллографа можно увидеть появление осциллограммы

Наконец, рис. 16.20 показывает процесс после завершения всех стадий изменения SF-диаграммы. Этот процесс означает устойчивое включение выключателя, так что теперь переменное напряжение подается все время на нагрузку — осциллограф. Выделение объектов SF-диаграммы прекращается.

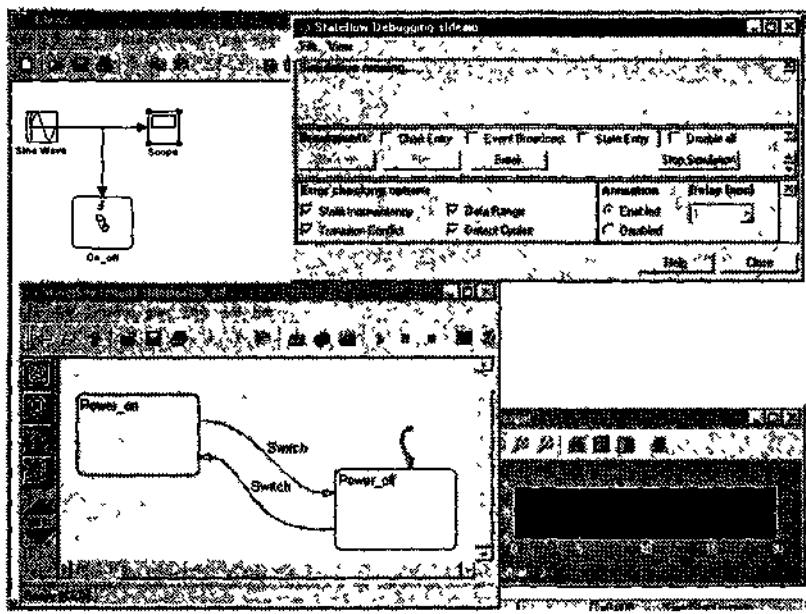


Рис. 16.18. Начало моделирования

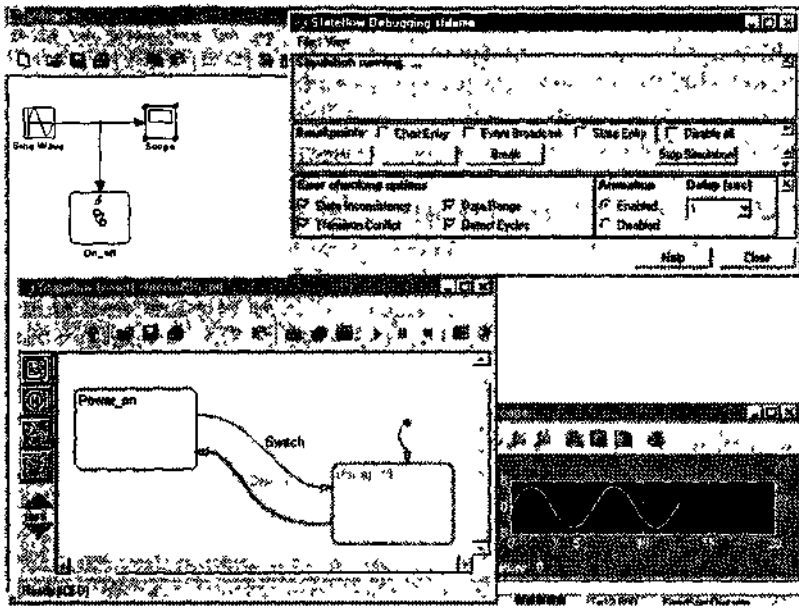


Рис. 16.19. Промежуточный кадр SF-диаграммы

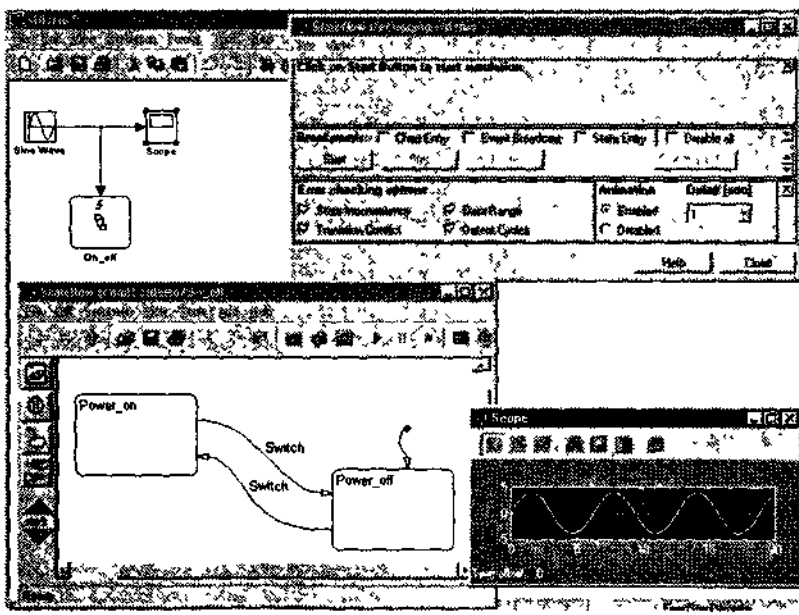


Рис. 16.20. Модель с SF-диаграммой в конце моделирования

Итак, нами выполнен весь цикл операций по составлению и запуску достаточно простой Simukink-модели выключателя со встроенной в модель SF-машиной и SF-диаграммой. Мы настойчиво советуем читателям провести этот опыт самостоятельно, прежде чем пытаться составлять 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.

Работа с простым отладчиком SF-диаграмм достаточно очевидна. Однако отметим несколько особенностей, которые полезно знать. Прежде всего это возможность выполнения моделирования по шагам с помощью кнопки Step. Анимацию SF-диаграмм можно отключить, установив в разделе Animation переключатель Disabled (анимация отключена). Наконец, можно замедлить стадии моделирования, установив нужное время задержки Delay (по умолчанию оно равно 1 с). Раздел Error Checking options позволяет отключить различные опции проверки ошибок. По умолчанию все они включены. Отключение этих опций целесообразно только после окончательной отладки SF-диаграммы.

Средства отладки SF-диаграмм

SF-модель, как программа системы MATLAB + Simulink, является типичной S-функцией. По составлению S-функций имеется специальное фирменное руководство. Однако визуально-ориентированное программирование, используемое в пакетах Simulink и Stateflow, позволяет без использования S-функций 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 меню Tools ▸ Parse редактора SF-диаграмм. Открывающееся при этом окно анализатора показано на рис. 16.21 в правом верхнем углу.

В этом окне имеется информация о текущей SF-диаграмме и наличии ошибок, если они есть. В нашем примере ошибок нет, поскольку он был отлажен. Однако следует иметь в виду, что анализатор

проверяет только синтаксические ошибки. Более каверзные семантические ошибки, например в выборе алгоритмов построения SF-диаграмм, не выявляются. Устранение таких ошибок — это дело пользователя.

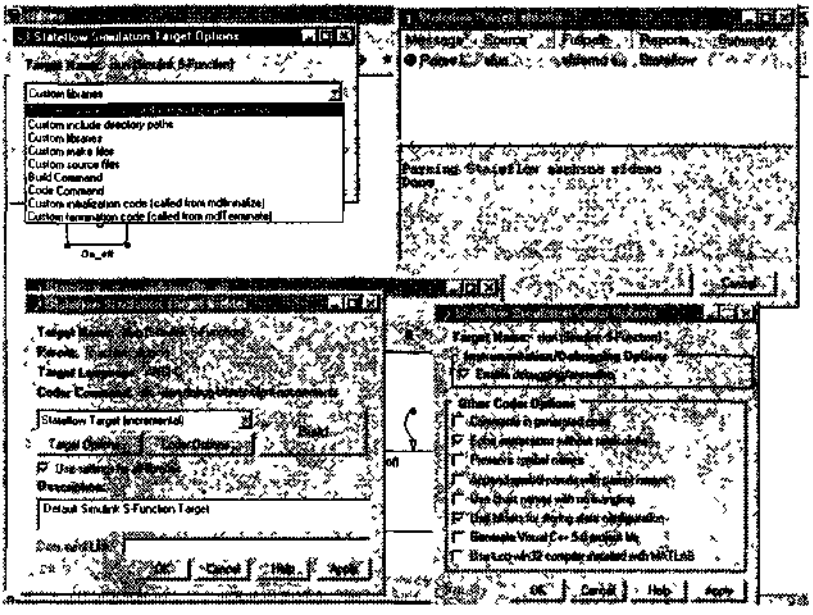


Рис. 16.21. Средства отладки SF-диаграмм

Для выделенного в окне анализатора сообщения можно открыть окно построителя целевого кода — Simulation Target Builder (см. рис. 16.21). Краткое описание вопросов генерации кодов будет дано в конце этой главы.

Кнопка Target Options открывает окно опций построителя целевого кода. На рис. 16.21 оно расположено в верхнем левом углу с открытым списком возможных опций. Другая кнопка — Coder Options — открывает окно опций специального компонента Stateflow — генератора программного кода Stateflow Coder (см. рис. 16.21). Здесь прежде всего надо отметить флажок Enable Debugging/Animation, который разрешает анимацию SF-диаграммы и по умолчанию включен. Ряд других флажков служит для управления процессом генерации кодов. Кнопка Build открывает окно, подобное окну Parse, но с именем Build в заголовке. Обычно это окно используется для представ-

ления сообщений о создавамой после нажатия кнопки Build DF-диаграмме

Таким образом, рис. 16 21 показывает все основные средства отладки SF-диаграмм. Как правило, эти средства нужны только в том случае, когда возникают проблемы с работой SF-диаграмм. Кроме того, они полезны и для обеспечения должного уровня надежности программ.

Поиск объектов SF-диаграмм

В SF-диаграммах большинства моделируемых систем, в том числе приведенных в демонстрационных примерах, очень сложно разобраться. Помощь в этом оказывает навигатор для поиска объектов. Он вызывается командой меню Tools ▸ Find... редактора SF-диаграмм. Эта команда открывает окно поиска объектов, показанное на рис. 16.22.

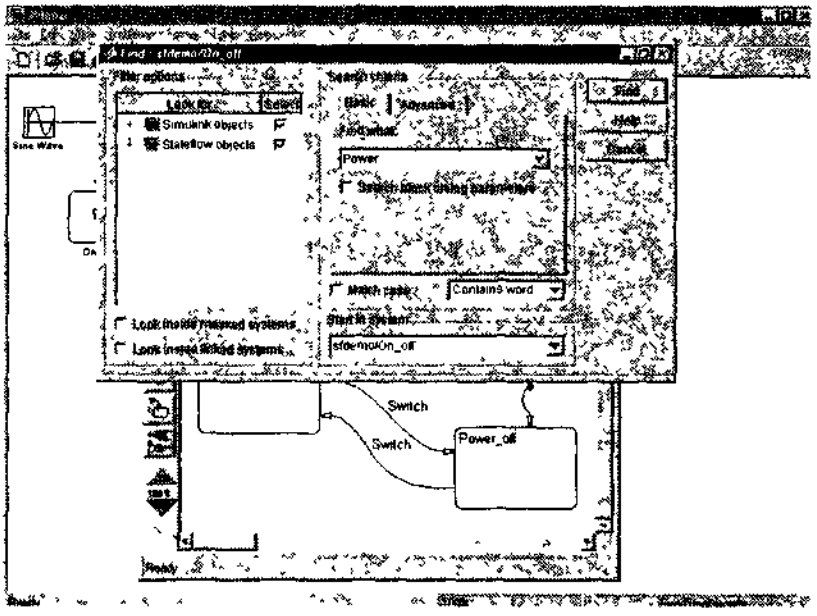


Рис. 16.22. Окно поиска объектов SF-диаграмм

В окне поиска можно задать имя объекта, который должен быть найден. Нажав кнопку Find, можно наблюдать результат поиска (рис. 16.23). Внизу окна появляется список всех объектов, в которых обнаружено заданное имя.

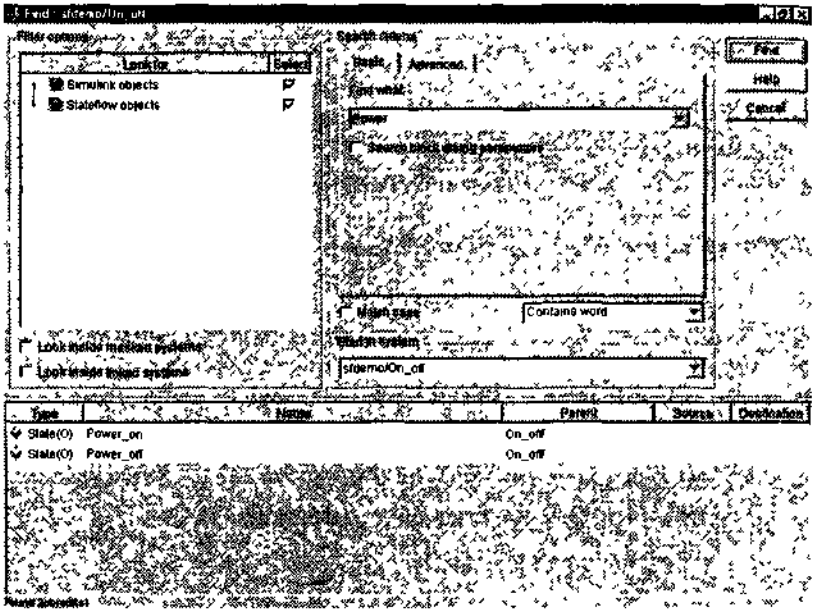


Рис. 16.23. Окно поиска объектов SF-диаграмм с обнаруженными блоками

Оформление SF-диаграмм

Выбор стиля SF-диаграмм

Размер объектов на SF-диаграммах можно менять с помощью переключателя под кнопками вывода графических объектов в панели инструментов (рис. 16.24). Возможно также задание разных стилей SF-диаграмм, отличающихся цветовым оформлением. Для этого в меню Edit есть команда Style..., которая выводит окно со схемами стиля диаграмм (см. рис. 16.24).

Можно выбрать стиль в меню, которое на рис. 16.24 представлено в открытом состоянии. Примеры задания стилей объектов имеются в окне выбора стиля.

Установка размера символов

Имеется также возможность установить размер символов для надписей SF-диаграммы. Для этого нужно воспользоваться командой

меню Edit ▸ Set Font Size окна редактора SF-диаграмм. В раскрывающемся меню остается выбрать нужный размер шрифта.

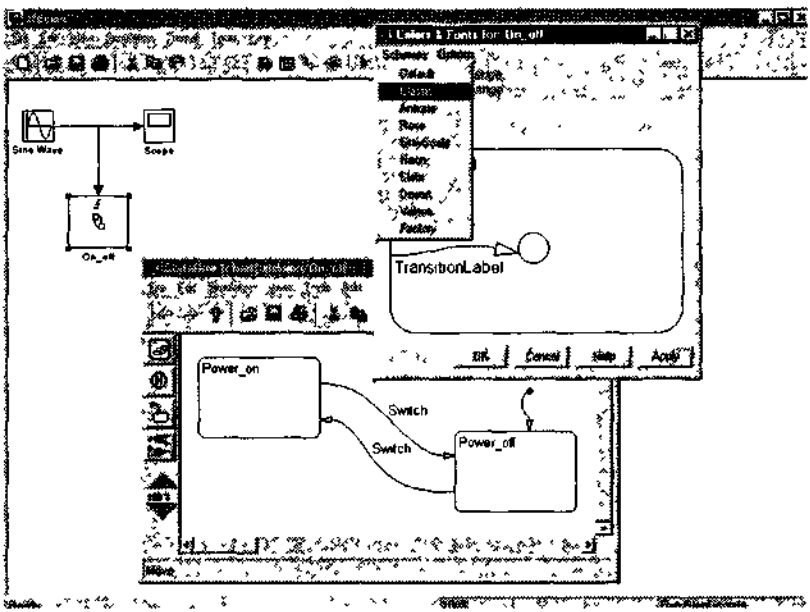


Рис. 16.24. Выбор стилей объектов SF-диаграмм

Демонстрационные примеры пакета Stateflow

16

Рекомендуемые правила работы

При ознакомлении с демонстрационными примерами пакета Stateflow с учетом их сложности (взаимодействие ряда систем) целесообразно придерживаться следующих рекомендаций:

1. Загрузите пример в окно пакета Simulink.
2. Ознакомьтесь с моделью примера и выясните наличие в ней субмоделей.
3. Просмотрите блок-схемы субмоделей и разберитесь с их работой.
4. Сделайте пробный запуск примера.

5. Если моделирование идет слишком долго, остановите его и проверьте установку параметров моделирования (в частности, конечного времени) и изменеите параметры.
6. Снова запустите пример на моделирование.
7. По завершении моделирования выведите осциллограммы и иные иллюстрации работы модели, предусмотренные в ней.
8. Постарайтесь осмыслить полученные результаты.
9. Выведите SF-диаграмму и запустите ее на исполнение (лучше делать это с помощью отладчика SF-диаграмм).
10. Проанализируйте SF-диаграмму, отметьте новые для вас детали и разберитесь с ними.
11. Сохраните результаты своего исследования.

Только очень наивный читатель может подумать, что стоит ему подготовить модель с SF-диаграммой и запустить ее в режиме моделирования, как можно сразу получать практически полезные результаты. На самом деле надо затратить немало часов для разбора как демонстрационных, так и 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 (как для знакомства с техникой моделирования в пакете Simulink, так и для применения пакета Stateflow) примеров. Мы рассмотрим их в порядке повышения сложности моделей.

Моделирование скользящего с трением бруска

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 сдвинется и начнет перемещаться. Если вектор силы периодически меняет на-

правление, то трение порождает гистерезис зависимости положения бруска от действующей на него силы.

Simulink-модель этой простой физической системы показана на рис. 16.25.

Там же представлены результаты моделирования — осциллограммы временной зависимости силы и перемещения бруска и фазовый портрет его движения, явно показывающий отмеченный выше гистерезис.

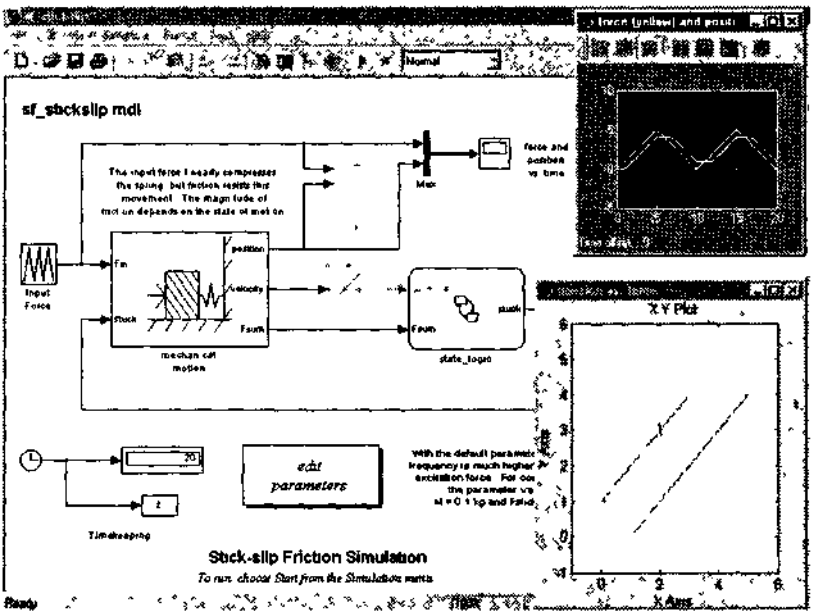


Рис. 16.25. Модель скользящего с трением бруска

SF-диаграмма этой модели также предельно проста, и это позволяет легко разобраться с ее особенностями. Она показана на рис. 16.26.

Моделирование поведения автомобиля

Рассмотрим еще один пример — моделирование поведения автомобиля. Соответствующая модель показана на рис. 16.27. Там же даны осциллограммы, показывающие поведение системы при возникновении на дороге чрезвычайного обстоятельства.

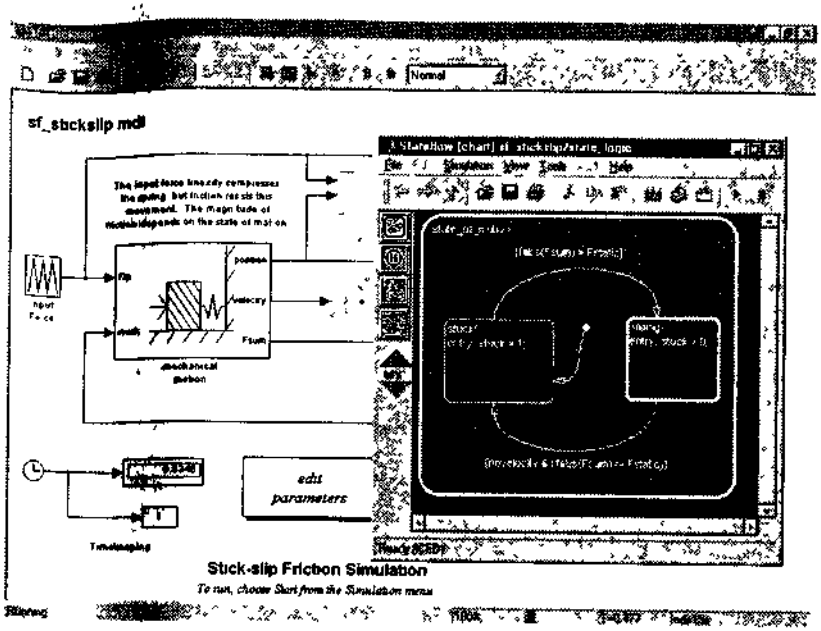


Рис. 16.26. SF-диаграмма модели скользящего с трением бруска

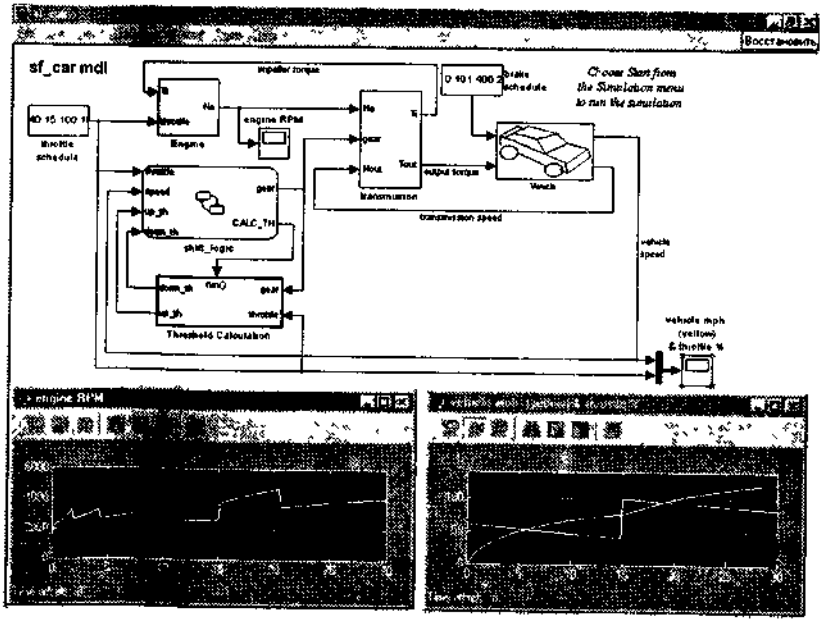


Рис. 16.27. Моделирование системы автомобиля

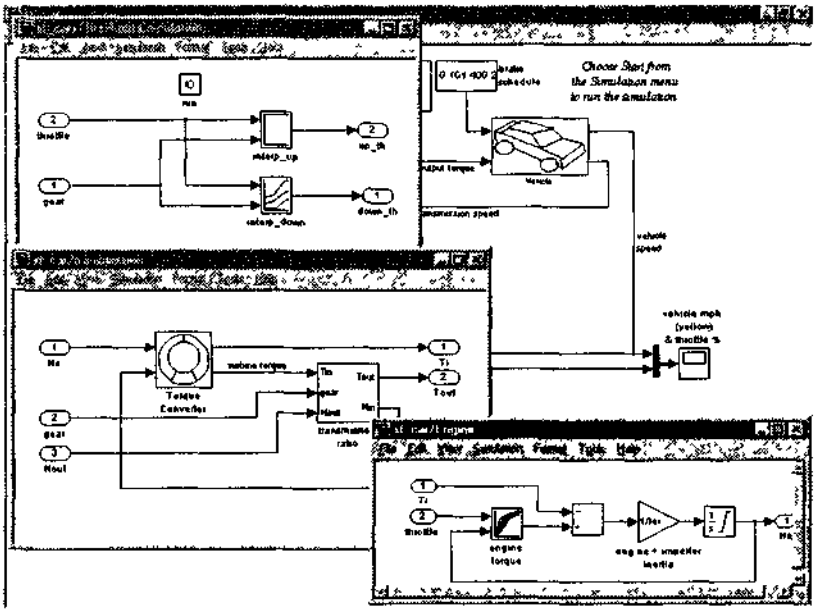


Рис. 16.28. Подсистемы модели поведения автомобиля

В отличие от рассмотренных ранее примеров эта модель имеет ряд субмоделей (рис. 16.28). Они содержат много интересных компонентов, демонстрирующих возможности системы Simulink. Несмотря на наличие субмоделей, этот пример все еще довольно прост и удобен для разбора.

Наконец, на рис. 16.29 представлена SF-диаграмма этой модели. Здесь прежде всего надо отметить наличие особого объекта — очереди, которая видна в верхней части SF-диаграммы. В очереди имеет место последовательная активизация одного блока вслед за другим под действием управляющих событий. В данный момент активен третий блок очереди.

Трудно себе представить более важную задачу, чем сохранение жизни водителя и пассажиров автомобилей при возникновении аварийных ситуаций на дорогах. Этот пример показывает полезность моделирования таких ситуаций. Мы намеренно оставляем за читателем более подробное рассмотрение этого и других примеров. Это позволит разобраться в деталях моделирования задач, представленных демонстрационными примерами.

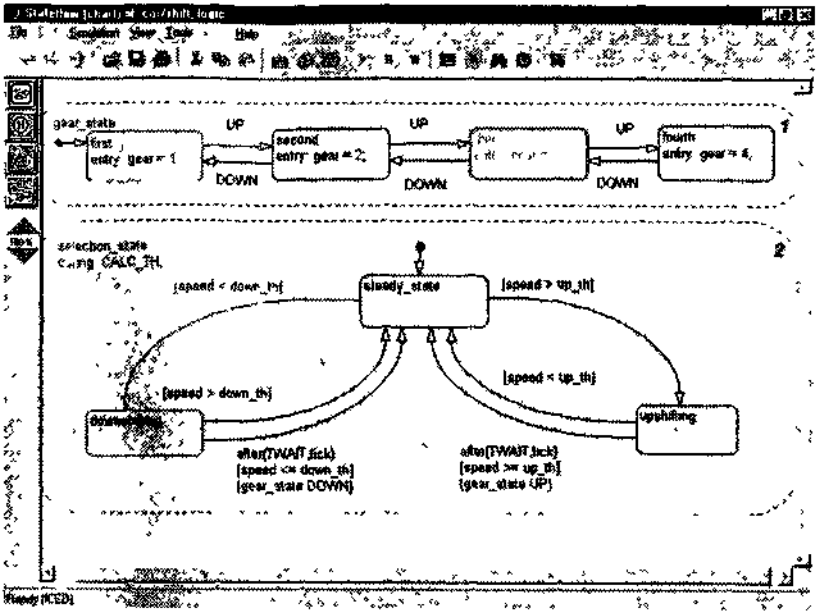


Рис. 16.29. SF-диаграмма модели поведения автомобиля

Моделирование электрогидравлического механизма

Еще один пример моделирования технического устройства представлен на рис.16.30. На этот раз моделируется электрогидравлический сервомеханизм. Такие механизмы широко применяются в промышленности и на транспортных средствах.

Вы можете самостоятельно ознакомиться с субмоделями этой модели. На рис. 16.31 показан пример применения обозревателя SF-диаграмм для этой модели. Чем сложнее модель, тем целесообразнее получение данных о ней с помощью обозревателя.

Рисунок 16 32 показывает применение отладчика SF-диаграмм для пошагового запуска SF-диаграммы. Это позволяет отдельно изучить каждый шаг диаграммы.

Следует отметить, что с помощью команд меню View окна отладчика можно задать различные размеры окна — на рис. 16.32 оно представлено в максимальных размерах. Минимальные размеры стоит устанавливать, если отладчик используется только для управления запуском SF-диаграммы, например в пошаговом режиме.

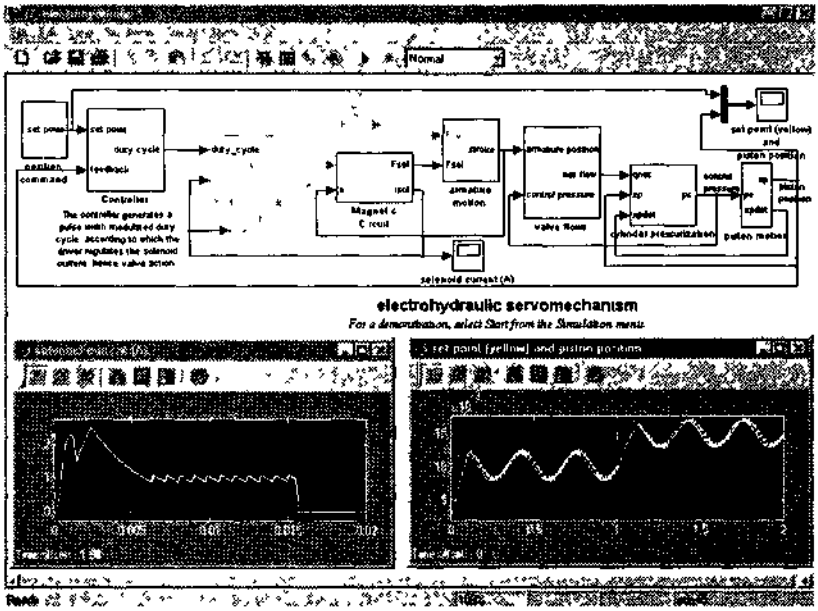


Рис. 16.30. Модель электрогигравлического сервопривода

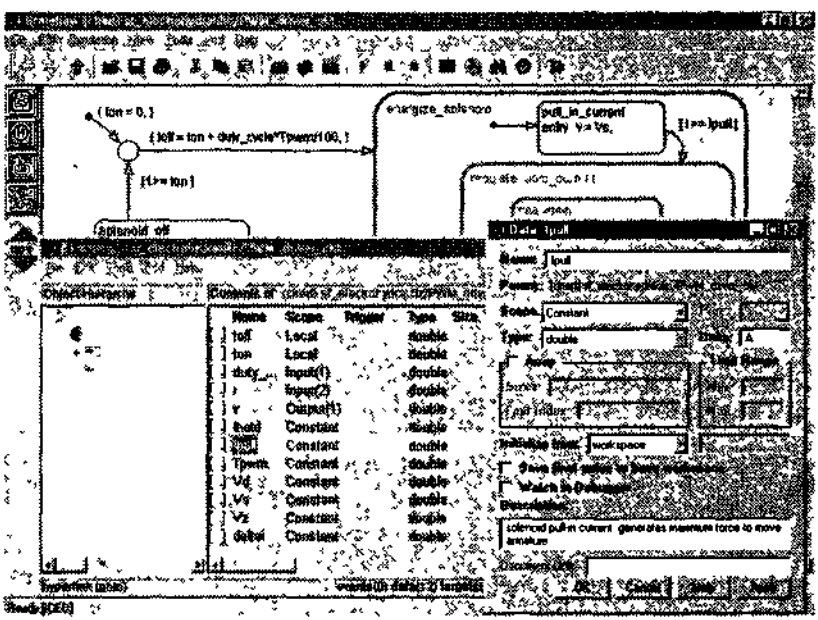


Рис. 16.31. Работа обозревателя SF-диаграмм с моделью электрогидравлического сервопривода

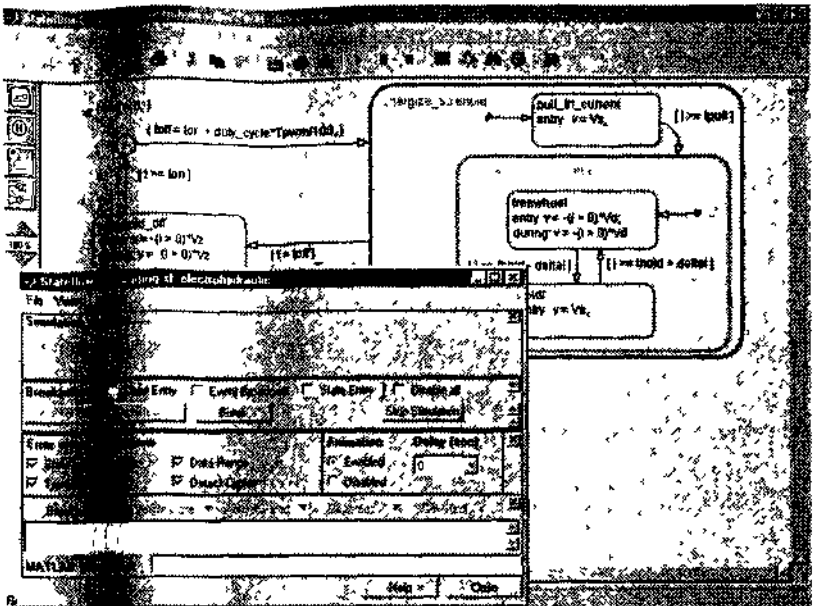


Рис. 16.32. Пошаговое выполнение SF-диаграммы электрогидравлического сервопривода

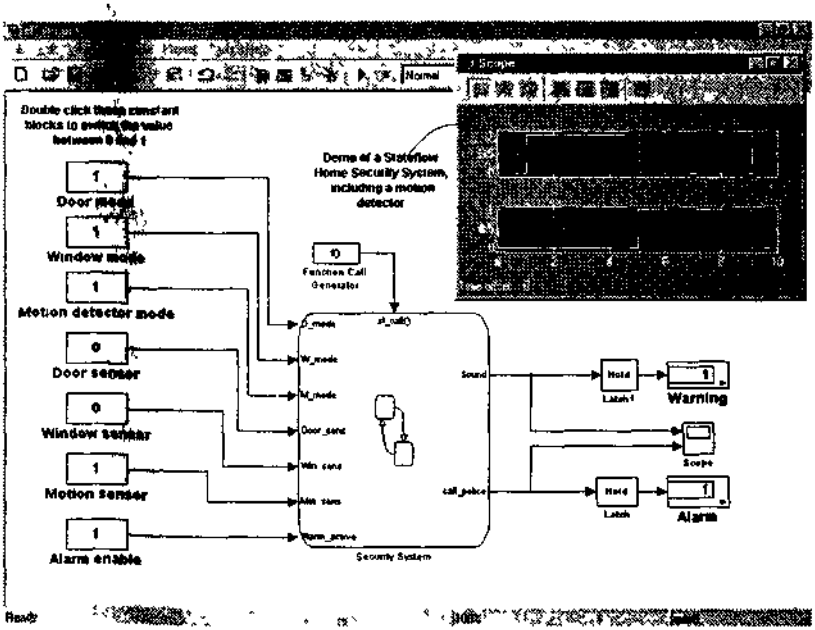


Рис. 16.33. Модель управления домашним хозяйством

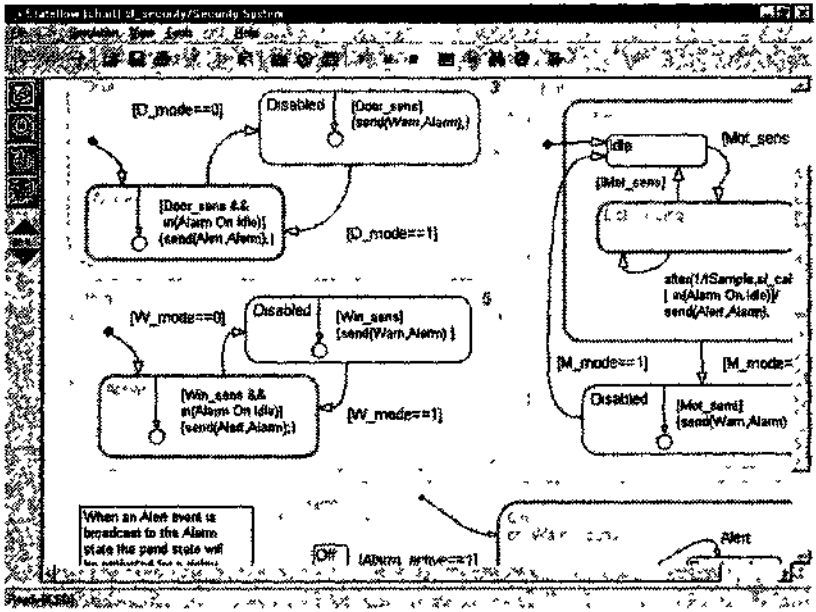


Рис. 16.34. Кадр работы SF-диаграммы системы на рис 16 33

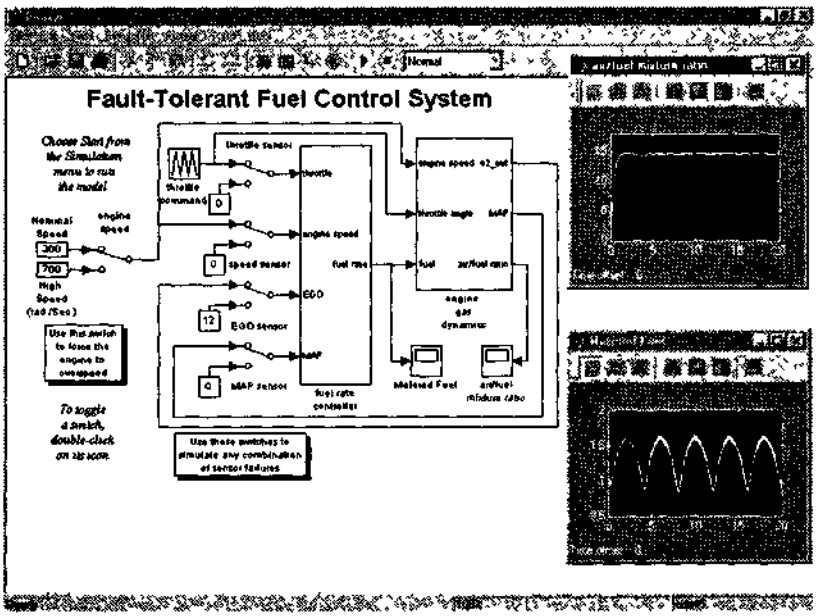


Рис. 16.35. Модель системы контроля топлива

Не забывайте и о возможности оперативно менять масштаб представления SF-диаграммы с помощью переключателя на левой стороне ее окна.

Моделирование системы управления домом

Еще одна интересная модель — система управления домашним «хозяйством» (дверями, окнами и т. п.) — представлена на рис. 16.33. В левой части задается (двойным щелчком мыши) состояние того или иного объекта. Можно похлопать дверью или окнами и посмотреть, что будет происходить в соответствии с логикой работы системы, заданной SF-диаграммой.

Ограничимся приведением кадра работы SF-диаграммы управления домашним хозяйством (рис. 16.34).

Моделирование отказоустойчивой системы контроля 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 контроля топлива. В демонстрационных примерах пакета Stateflow есть пример такого рода — fuelsys. Рисунок 16.35 показывает модель после ее запуска и вывода контрольных осциллограмм.

Эта модель также содержит множество субмоделей. Рисунок 16.36 дает лишь представление об этом. При этом стоит отметить, что в данном примере субмодели в поле рисунка уже не вмещаются, да и показаны они не все.

Кадр работающей SF-диаграммы представлен на рис. 16.37. Здесь также можно найти множество интересных примеров задания описаний различных объектов SF-диаграммы, в частности ее состояний и управляемых событиями переходов.

Этими примерами мы закончим рассмотрение очень полезной и интересной системы событийного моделирования Stateflow. Надо полагать, что читатель отдаст должное уникальным возможностям этого пакета и самостоятельно ознакомится с другими демонстрационными примерами на построение SF-диаграмм.

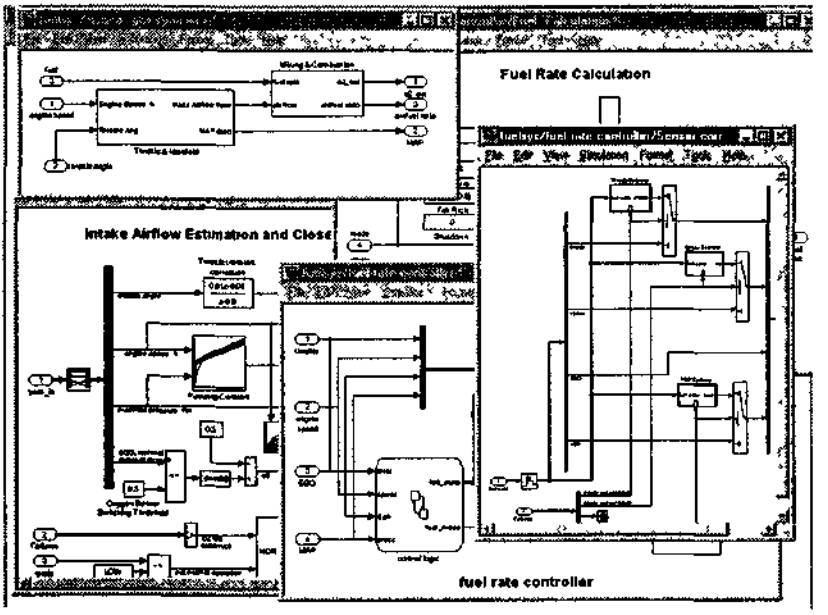


Рис. 16.36. Пример вывода некоторых субмоделей для модели рис. 16 35

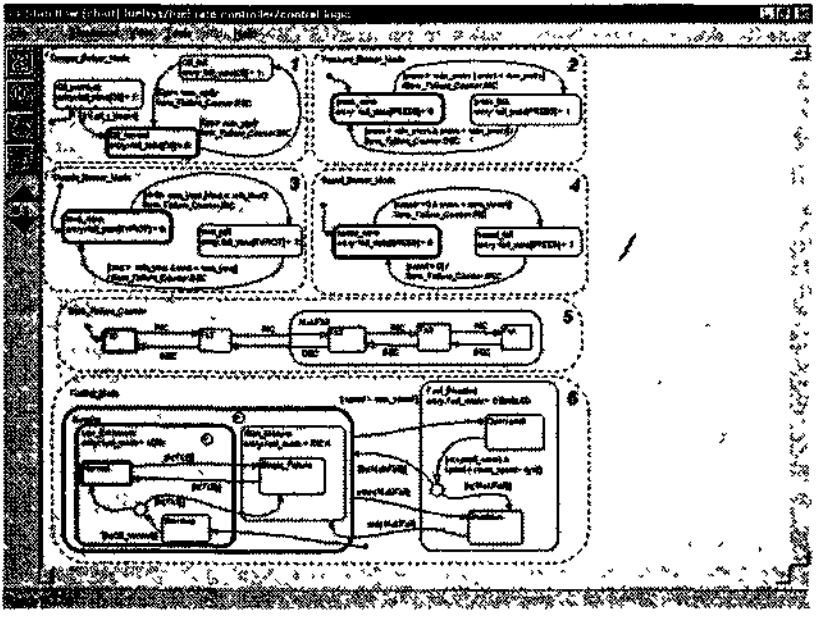


Рис. 16.37. SF-диаграмма системы контроля топлива

Мастерская реального времени Real-Time Workshop

Назначение пакета Real-Time Workshop (RTW)

Пакет расширения Real-Time Workshop (RTW) – это мастерская 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 компоновка программных модулей с последующей компиляцией. Эти файлы обычно используются для управления внешними устройствами, работающими в реальном масштабе времени, например роботами или компьютеризованными измерительными комплексами.

Процесс создания исполняемых кодов моделей имеет некоторые ограничения. Так, в состав компилируемых моделей 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 нелинейных систем. И наконец, не стоит забывать о необходимости установки самого пакета и обеспечения его бесконфликтной работы с компилятором.

Кроме того, на ПК должен быть установлен один из следующих компиляторов:

Платформа	Компилятор	Файл шаблона
Microsoft Windows	MS Visual C/C++	grt_vc tmf
Microsoft Windows	Borland C/C++	grt_bcx.tmf
Microsoft Windows	Watcon C	grt_watc.tmf
UNIX	Любой ANSI UNIX C-компилятор	grt_unix.tm

Работа пакета RTW

Рассмотрим работу пакета RTW. 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 любой из описанных в данной книге. Этот файл вместе с системным файлом — шаблоном system.tmf поступает в утилиту Build пакета RTW. Эта утилита создает промежуточный файл model.rtw в независимом от языка формате ASCII. Последний содержит все необходимые для функционирования модели данные, например значения параметров, типы и размерности сигналов, дискретность модельного времени и т. д.

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 следующего назначения:

- model.c — задает 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).

Исполняемый файл может быть вызван как из командной строки системы MATLAB, так и без нее — средствами операционной системы. Набор созданных пакетом RTW-файлов размещается в папке WORK.

Что дает компиляция моделей

Заметим, что выполнение моделей в системе MATLAB + Simulink происходит обычно в интерпретирующем режиме. Это означает, что каждая команда или функция сначала анализируется и только после этого выполняется. Это ведет к относительно медленной работе системы, хотя даже как интерпретатор MATLAB выгодно отличается от других систем компьютерной математики. Это достигнуто усовершенствованием матричных алгоритмов, используемых в MATLAB.

Компиляция означает, что этапы интерпретации и компоновки при исполнении файлов отсутствуют, поскольку проводятся в процессе создания исполняемых файлов. В результате скорость вычислений при использовании исполняемых (откомпилированных) файлов намного возрастает — до нескольких десятков раз. Это позволяет заметно ускорить моделирование типовых систем, особенно когда нужно проводить его многократно с изменением параметров. При использовании системы MATLAB для управления и контроля внешних систем и устройств, работающих в реальном масштабе времени, это просто жизненно необходимо.

Заметим, что в подавляющем большинстве системы компьютерной математики не позволяют создавать исполняемые в реальном масштабе времени файлы своих моделей, что вынуждает применять их файлы для работы только при загруженной 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 их совместно с внешними программами и внешним оборудованием, которое может быть подключено к ПК.

Однако не стоит пренебрегать трудностями такого подхода. Помимо уже указанных в этой главе трудностей стоит упомянуть самые разные виды нестыковки как между аппаратными, так и между программными компонентами систем, на аппаратном уровне работающих в реальном масштабе времени. Поэтому описание технологии проектирования и практической отладки таких систем далеко выходит за рамки этой книги, а приведенный ниже материал является лишь ознакомительным.

Работа пакета RTW с внешними устройствами

Файл model.exe используется с платами расширения, имеющими встроенное программное обеспечение. Его коды используются для старта пакета Simulink в режиме внешнего управления (см. пункт External меню Simulation окна Simulink). Так образуется (когда это необходимо) замкнутая система управления внешними устройствами. Она обладает большими возможностями программного управления и обработки поступающей из них информации, которые пре-

доставляет программный комплекс MATLAB + Simulink + Пакеты расширения

Режим внешнего управления позволяет осуществлять динамическую настройку параметров исполнения, созданных пакетом RTW кодов на основе применения механизма API (Application Program Interface — интерфейс прикладных программ), имеющегося у операционных систем класса Windows 95/98/NT Получаемая при этом среда разработки программного кода имеет название среды быстрого макетирования (rapid prototyping program framework)

Кратко работа MATLAB + Simulink в режиме внешнего управления происходит следующим образом Сначала при соответствующих исходных значениях параметров выполняется моделирование и строится SF-диаграмма После этого Simulink переходит в ждущий режим и ожидает модификации параметров Модификация параметров может быть выполнена как пользователем в среде Simulink, так и по сигналам от внешнего оборудования Опуская описание самого механизма передачи, отметим, что таким образом осуществляется замкнутый цикл управления моделью с возможностью вмешательства оператора

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 или отдельных ее блоков,
- параметры блоков типа Fcn (функции)

Иными словами, можно моделировать лишь системы с неизменяемой структурой Тем не менее пакет RTW (а также ряд других пакетов) позволяет вести эффективную работу с самыми разными периферийными устройствами, такими как устройства обработки сигналов с различных датчиков, устройства контроля напряжений, токов и других сигналов, аппаратные средства для управления роботами (в том числе промышленными и бытовыми) и многими другими устройствами

Как правило, все они требуют приобретения довольно специфичных и дорогих аппаратных средств (плат расширения ПК), стоимость

которых нередко намного превышает стоимость современного ПК с установленным на нем базовым пакетом MATLAB + Simulink Но и возможности такого применения системы MATLAB становятся почти фантастическими! Здесь эпитет «почти» употреблен только потому, что многие из этих устройств уже есть на практике, а другие создаются во многих лабораториях мира

Подготовка к созданию исполняемого файла

Для создания исполняемого файла в нормальном режиме моделирования необходимо прежде всего запустить MATLAB + Simulink и те пакеты, которые нужны для создания компилируемой модели Далее следует просмотреть параметры модели (как это делается, описывалось выше) и параметры пакета RTW Для этого выберите пункт меню Tools ▸ Parameters... пакета Simulink и в появившемся окне параметров откройте вкладку Real Time Workshop

Запуск компиляции производится нажатием кнопки Build на этой вкладке Можно также сделать это из окна параметров пакета Stateflow RTW Target Builder (большая кнопка RTW Build) В процессе компиляции на передний план выводится окно системы MATLAB, в котором можно проследить за созданием файлов в ходе компиляции

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) Это также возможно, если в файле autoexec.bat не будет указан доступ в папку компилятора

При успешной компиляции исполняемый файл и сопровождающие его файлы размещаются в папке WORK, расположенной в папке системы MATLAB Вместе с exe-файлом создается пусковой файл с расширением .bat и файл на языке C с расширением .c Исполняемые файлы можно вызвать в командной строке системы MATLAB в формате !Имя_файла Символ ! означает, что MATLAB передает управление операционной системе Командой load имя_файла можно сохранить результаты Здесь нужно ввести то имя файла, которое было выбрано для исполняемого файла

В заключение следует отметить, что далеко не каждая модель сразу компилируется Как правило, большинству моделей требуется до-

работка, прежде чем компиляция пройдет успешно. Отчасти это связано с ограничениями, налагаемыми на параметры модели перед компиляцией. Они были отмечены выше. Например, 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.

К тому же и успешная компиляция вовсе не означает, что исполняемый файл будет корректно работать. Правильнее использовать полученный на языке С файл модели для его последующей отладки и компиляции в компиляторе языка C/C++. Таким образом, создание корректно работающих исполняемых файлов все еще остается уделом опытных программистов, и вряд ли эту работу стоит рекомендовать начинающим пользователям. Для них гораздо более привычной 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 весьма специфических внешних устройств, так и деталей сетевых протоколов, на которых они базируются. Для полноты обзора средств RTW отметим, что в его состав может включаться расширение RTW Ada Coder, позволяющее создавать коды на языке Ada (стандарт Ada95).

Литература

1. *Дьяконов В. П.* Компьютерная математика: Теория и практика. М.: Нолидж, 2001.
2. *Дьяконов В. П.* MATLAB: Учебный курс. СПб.: Питер, 2001.
3. *Дьяконов В., Новиков Ю., Рычков В.* Компьютер для студента: Самоучитель. СПб.: Питер, 2000.
4. *Дьяконов В., Круглов В.* Математические пакеты расширения MATLAB: Специальный справочник. СПб.: Питер, 2001.
5. *Дьяконов В. П., Абраменкова И. В.* MATLAB 5.0/5.3. Система символьной математики. М.: Нолидж, 1999.
6. *Дьяконов В. П.* Расширяемые системы для численных расчетов MatLAB / Монитор-Аспект. 1993. № 2.
7. *Дьяконов В. П.* Справочник по применению системы PC MatLAB. М.: Наука; Физматлит, 1993.
8. *Лазарев Ю.* MatLAB 5.0. Библиотека студента. Киев: Ирина, BHV, 2000.
9. *Потемкин В. Г.* MATLAB 5 для студентов. М.: Диалог-МИФИ, 1998.
10. *Потемкин В. Г.* Система инженерных и научных расчетов MATLAB 5.x: В 2 т. М.: Диалог-МИФИ, 1999.
11. *Потемкин В. Г.* Инструментальные средства MATLAB 5.X. М.: Диалог-МИФИ, 2000.
12. *Медведев В. С., Потемкин В. Г.* Control System Toolbox. MATLAB 5 для студентов. М.: Диалог-МИФИ, 1999.
13. *Гультяев А.* MATLAB 5.2. Имитационное моделирование в среде Windows. СПб.: КОРОНА-принт, 1999.
14. *Гультяев А.* Визуальное моделирование в среде MATLAB. Учебный курс. СПб.: Питер, 2000.
15. *Рудаков П. И., Сафонов В. И.* Обработка сигналов и изображений. MATLAB 5.x. / Под общей редакцией к. т. н. В. Г. Потемкина. М.: Диалог-МИФИ, 2000.
16. *Мартынов Н.Н., Иванов А. П.* MATLAB 5.x: Вычисления, визуализация, программирование. М.: КУДИЦ-ОБРАЗ, 2000.

Алфавитный указатель

Символы

A

B

C

D

E

F

Q

R

S

T

U

V

W

Z

А

Б

В

Г

Д

З

И

К

Л

M

Н

О

П

Р

С

Т

У

Ф

Х

Ц

Ч

Ш

Э