В. П. Дьяконов

Simulink 5/6/7

Самоучитель

ДМК Пресс
издательство
Москва, 2008

УДК 32.973.26-018.2
ББК 004.438
Д93

Д93 **Дьяконов В. П.**
Simulink 5/6/7: Самоучитель. – М.: ДМК-Пресс, 2008. – 784 с.: ил.

ISBN 978-5-94074-423-8

Самоучитель по новейшим реализациям пакета визуального блочного имитационного моделирования Simulink 5/6/7 матричной системы MATLAB R2006/2007. Подробно описаны библиотека блоков Simulink, методика подготовки диаграмм моделей, их редактирование, настройка и запуск на исполнение. Дано описание наиболее важных пакетов расширения Simulink инструментального ящика Blockset, в том числе SimPowerSystems, SimMechanics, Aerospace, Stateflow, Signal Processing, Telecommunication, Video and Image Processing и др. Отражены средства виртуальной реальности. Описаны сотни наглядных примеров применения этих средств.

Издание предназначено для студентов, преподавателей и аспирантов вузов и университетов, инженеров и научных работников.

УДК 519.6
ББК В162я73

ISBN 978-5-94074-423-8

Краткое содержание

Содержание

Введение

В наши дни компьютерная математика получила должную известность и находится на этапе быстрого развития [1]. Начав свой путь с применения программируемых микрокалькуляторов и расчетов на персональных ЭВМ (компьютерах), она породила целый ряд своих специальных программных средств – систем компьютерной математики (СКМ) и пакетов их расширения.

Среди СКМ, в первую очередь ориентированных на численные расчеты, особо выделяется матричная математическая система MATLAB – самая эффективная среди систем для численных вычислений. Система фактически стала мировым стандартом в области современного математического и научно-технического программного обеспечения. В разработке MATLAB и пакетов расширения этой системы принимают участие крупные научные школы мира, многие ведущие университеты и иные организации.

Эффективность MATLAB обусловлена прежде всего ее ориентацией на матричные вычисления [2, 3] с программной эмуляцией параллельных вычислений и упрощенными средствами задания циклов. В MATLAB удачно реализованы средства работы с многомерными массивами, большими и разреженными матрицами и многими типами данных. Система прошла многолетний путь развития от узко специализированного матричного программного модуля, используемого только на больших ЭВМ, до универсальной интегрированной СКМ, ориентированной на массовые персональные компьютеры класса IBM PC, AT и Macintosh и рабочие станции UNIX. MATLAB имеет мощные средства диалога, графики и *комплексной визуализации* вычислений, в том числе с привлечением средств дескрипторной графики и виртуальной реальности.

Система MATLAB предлагается разработчиками (корпорация The MathWorks, Inc.) как лидирующий на рынке, в первую очередь на предприятиях военно-промышленного комплекса, в энергетике, в аэрокосмической отрасли и в автомобилестроении, язык программирования *высокого уровня* для технических вычислений, расширяемый большим числом пакетов прикладных программ – *расширений*. В новых реализациях их число составляет сотни, причем свыше 80 пакетов расширений фирмы MathWorks входят в полную поставку новейшей системы MATLAB R2007b. Самым известным из них стало расширение Simulink, обеспечивающее блочное имитационное моделирование различных систем и устройств с применением современной технологии визуально-ориентированного программирования.

Структура комплекса MATLAB + Simulink, помимо основы – системы MATLAB и главного расширения Simulink, содержит обширные группы пакетов расширения (рис. 0.1). Это инструментальные ящики Toolboxes с большим числом пакетов расширения, приближающимся к сотне, и группа пакетов расширения Blocksets, увеличивающая возможности системы визуально-ориентированного блочного имитационного моделирования динамических систем Simulink.

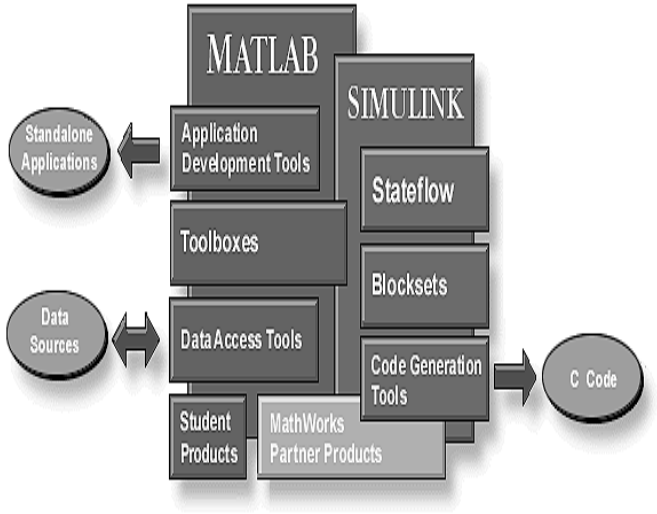


Рис. 0.1. Структура системы MATLAB + Simulink

В России первой книгой по системе MATLAB стала книга [4], выпущенная еще в 1993 г. В дальнейшем было опубликовано множество книг по различным версиям этой мощной системы [5–49]. Так, только на Web-узле корпорации The MathWorks, Inc., разработавшей эту систему, указано уже более 1000 книг. Среди них есть и книги автора (рис. 0.2), вошедшие в программу поддержки подготовки книг корпорации The The MathWorks, Inc. (далее просто The MathWorks).

К сожалению, в России мало публикаций по новейшим версиям системы MATLAB и особенно по ее расширению Simulink. Лишь после 1997 г. появился ряд книг по системам MATLAB и отдельным пакетам расширения этой системы [4–43]. Из них следует особо отметить трехтомник автора [13–15] по версиям MATLAB 6.* и пятитомник по MATLAB 6.5 и MATLAB 7 [16–20]. Предшествующие Simulink 6.* версии пакета Simulink описаны в книгах [6, 9, 13, 16, 32–34], из которых только две [9, 33] посвящены отдельно расширению Simulink. По новым версиям Simulink 6.* Simulink 7 литература у нас отсутствует.

Увы, но объем книг по системе MATLAB и ее пакетам расширения так же непрерывно растет, как и их стоимость. Достаточно отметить, что последние книги только по системе MATLAB учебного характера имеют объем более 750 [29] и даже свыше 1100 [28] страниц. Пять последних томов автора по системе MATLAB с пакетами расширения насчитывают уже более 2800 страниц [16–20].

Фирменная документация по системе (англоязычная) представлена многими десятками книг, например [44–49]. Она настолько разрослась, что разработчики MATLAB были вынуждены прекратить поставки ее в виде PDF-файлов на отдельном CD-ROM или DVD и ныне разместили ее на своем интернет-сайте. Однако из-за большого объема файлов документации скачать их весьма проблематично даже для тех наших пользователей, которые имеют доступ в обычный

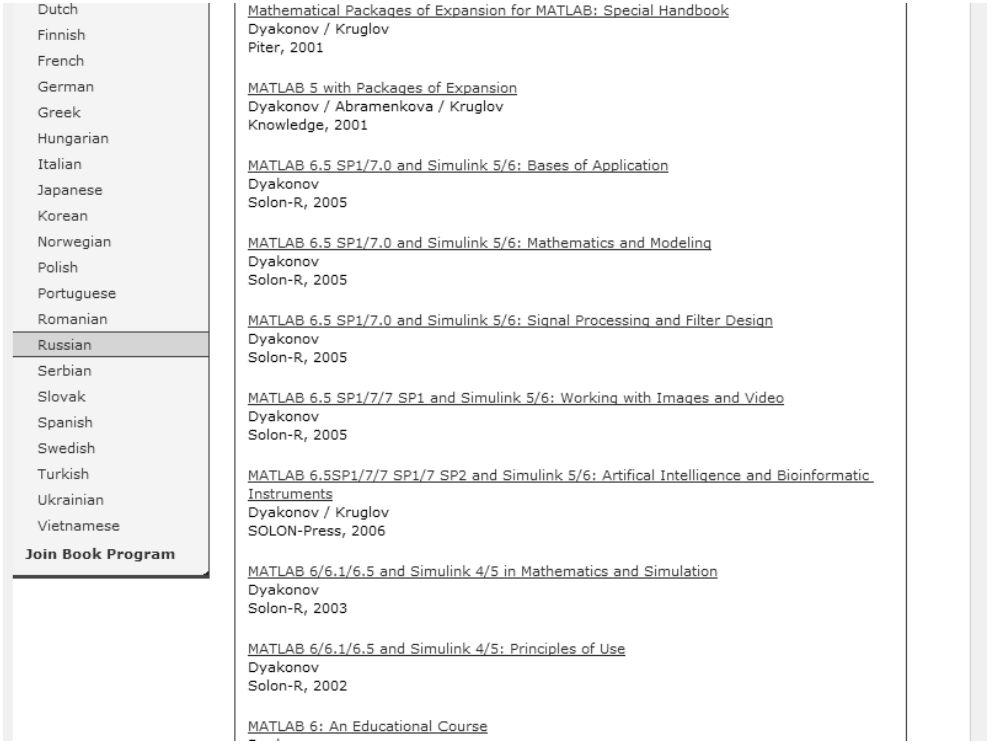


Рис. 0.2. Книги автора по системе MATLAB и пакетам ее расширения на интернет-сайте корпорации The MathWorks

Интернет. Кроме того, информация в PDF-файлах англоязычная и огромная по размеру.

Все это делает книги по MATLAB и Simulink доступными лишь для малой части наших инженеров, научных работников и преподавателей вузов и университетов. Достаточно отметить, что стоимость всего одной крупной книги по системе MATLAB составляет примерно трехмесячную выплату на приобретение литературы, предусмотренную в государственных вузах и университетах Российской Федерации для их преподавателей.

В связи с этим актуальной стала подготовка серии относительно небольших учебных курсов и самоучителей по системе MATLAB и ее приложениям. Однако в связи с отсутствием в программах большинства вузов и университетов учебных курсов по системе MATLAB более целесообразной является подготовка самоучителей, ориентированных на широкую читательскую аудиторию: студентов, аспирантов и преподавателей вузов и университетов, инженеров и научных работников. Все они вынуждены осваивать MATLAB и пакеты расширения этой системы самостоятельно.

Самоучитель по базовой системе MATLAB уже был предложен читателям. В данном, втором самоучителе впервые описаны новейшие реализации пакета

расширения Simulink 5/6/7 систем MATLAB 6.5 (R13)/7(R14)/R2006/R2006a/ R2006b/R2007a/R2007b.

Simulink – главный пакет расширения системы MATLAB, реализующий имитационное блочное визуально-ориентированное моделирование систем и устройств как самого общего, так и конкретного назначения. В книге описаны и современные реализации наиболее важных пакетов расширения Simulink инструментального ящика Blockset: Simulink Response Optimizarion для моделирования и оптимизации отклика нелинейных систем, SimPowerSystems для моделирования и проектирования энергетических систем и устройств, SimMechanics для моделирования и проектирования механизмов и механических систем, Signal Processing для моделирования систем обработки сигналов, Telecommunication для моделирования систем телекоммуникаций, Video and Image Processing для моделирования систем, работающих с видеопотоками и видеоизбражениями, и др. Описаны средства виртуальной реальности MATLAB.

Отличительными особенностями данной книги являются:

- достаточно полное теоретическое введение по основам математического моделирования различных систем и устройств;
- особое внимание к технике наглядной графической визуализации математического моделирования, в том числе на основе применения средств виртуальной реальности пакета Virtual Reality Toolbox;
- ориентация на современное визуально-ориентированное блочное моделирование, реализованное в новейших версиях пакета расширения системы MATLAB + Simulink;
- ориентация на читателей, желающих самостоятельно освоить технику визуально-ориентированного математического блочного моделирования на основе базовой системы MATLAB и ее расширений для такого моделирования – прежде всего Simulink 6/7;
- компактность книги при сохранении достаточной полноты изложения материала и справочных данных;
- описание новейших реализаций пакета расширений Simulink 6/7 для систем MATLAB R2006/R2006a/R2006b/R2007a/ R2007b;
- выделение (прямо скажем, не очень больших и глубоких) отличий между реализациями Simulink 5/6/7;
- более систематизированное и глубокое изложение материала, в том числе относящегося к библиотекам блоков пакета расширения Simulink и других пакетов расширения;
- значительно более подробное, чем в прежних книгах автора, и собранное воедино описание новейших реализаций пакетов расширения Simulink: Nonlinear Control Design, SimPowerSystems, SimMechanics и Aerospace;
- доступность основного материала пользователям версий MATLAB Simulink 5.*;
- разделение материала книги на отдельные уроки, каждый из которых, в зависимости от глубины изучения материала, может потребовать от 4 до 6 академических часов.

Внедрение системы MATLAB + Simulink в учебный процесс вузов России и стран СНГ находится в начальной стадии. В большинстве наших вузов и университетов пока нет отдельного курса по этой системе, но спецкурсы по ней уже появились. Довольно часто изучение MATLAB выполняется в рамках курсов по численным методам вычислений и математическому моделированию. В связи с этим особенно важным представляется издание самоучителей по системе MATLAB и пакетам ее расширения, часто охватывающим целые направления науки и техники.

Данная книга может служить не только самоучителем по расширению Simulink системы MATLAB и ряду относящихся к нему других расширений, но и достаточно полным учебным курсом, учебным пособием и даже справочником по Simulink. Книга предполагает знакомство пользователя с базовой системой MATLAB, например в пределах самоучителя по этой системе (первой книги данной серии). Однако в целом книга носит вполне самостоятельный характер.

Благодарности и адреса для связи

Автор выражает благодарность представителям корпорации The The MathWorks, Inc. Courtney Esposito, Naomi Fernandes и Meg Vulliez. Благодаря им подготовка автором книг по системе MATLAB и ее расширениям уже многие годы включается в планы поддержки этой корпорацией издания книг по системе MATLAB в разных странах мира и обеспечивается самыми свежими лицензионными программными средствами с обширной документацией по ним.

Доктору технических наук, профессору Владимиру Круглову и кандидату физико-математических наук, доценту Роману Кристалинскому автор выражает признательность за постоянный обмен мнениями и просмотр некоторых материалов этой книги. Автор благодарит также Генерального директора ЗАО «Смоленский Телепорт» (www.keytown.com) Григория Рухамина за предоставление услуг Интернета в ходе работы над книгой, что позволило посредством прямой оперативной связи с сайтом фирмы The MathWorks, Inc., быть в курсе обновлений системы MATLAB и использовать самую свежую информацию.

С автором можно связаться по электронной почте (vpdyak@keytown.com). Автор заранее выражает признательность всем читателям, которые готовы сообщить свое мнение о данной книге и поделиться соображениями по ее улучшению. Кроме электронной почты, замечания можно направлять по следующему адресу: 214000, г. Смоленск, ул. Пржевальского, 4, СГПУ. Вы можете отправлять свои письма и по адресу издательства, выпустившего книгу.

Связаться с фирмой The MathWorks вы можете, посетив сайт www.mathworks.com. Ее официальные почтовые реквизиты следующие:

The The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA, 01760-2098 USA
Tel: 508-647-7000
Fax: 508-647-7101
E-mail: info@The MathWorks.com

Урок 1

Краткое введение в математическое моделирование

Математическое моделирование основано на достижениях математики – как классической, так и новейшей компьютерной, ориентированной на выполнение вычислений с помощью современных компьютеров [1–40, 66–69]. Чтобы работать с такой мощной системой математического моделирования, как Simulink, нужен определенный минимум теоретических знаний по математике, численным методам [50–64] и математическому моделированию [71–81]. Он и содержится в данном уроке.

1.1. Основные понятия моделирования

1.1.1. Значение математического моделирования

Моделирование можно рассматривать как замещение исследуемого объекта (оригинала) его условным образом, описанием или другим объектом, именуемым *моделью* и обеспечивающим адекватное с оригиналом поведение в рамках некоторых допущений и приемлемых погрешностей. Моделирование обычно выполняется с целью познания свойств оригинала, путем исследования его модели, а не самого объекта. Разумеется, моделирование оправдано в том случае, когда оно проще создания самого оригинала или когда последний по каким-то причинам лучше вообще не создавать. Вопросам моделирования посвящена обширная литература, отдельно отметим работы [67–69, 71–81].

Исключительно велика роль моделирования в ядерной физике и энергетике. Достаточно сказать, что замена натурных ядерных испытаний моделированием не только экономит огромные средства, но и благоприятно сказывается на экологии планеты Земля. А такое явление, как «ядерная зима», вообще может исследоваться только на моделях, поскольку произойди оно на самом деле, это означало бы уничтожение жизни на Земле. Запрет на испытания ядерного оружия стал возможен также благодаря самым изысканным средствам моделирования ядерных и термоядерных процессов. Трудно переоценить роль моделирования в космонавтике и авиации, в предсказании погоды, в разведке природных ресурсов и т. д.

Однако не только такие показательные примеры демонстрируют роль математического (и компьютерного) моделирования. На самом деле моделирование даже самых простых и широко распространенных устройств, например работы сливного бачка в туалете или электрического утюга, ведет к огромной экономии средств и улучшению качества массовых изделий. Чем сложнее проектируемый объект, тем, как правило, важнее роль моделирования в его изучении и создании. Самое широкое применение моделирование находит в механике и физике, электротехнике, радиотехнике и электронике, в технике обработки сигналов и коммуникаций. В свою очередь, успехи в этом направлении способствуют созданию аппаратных и программных средств математического моделирования.

Трудно переоценить роль моделирования в образовании, где нередко реальные дорогие лабораторные работы приходится заменять компьютерным моделированием. Но, пожалуй, главное заключается в том, что математическое моделирование позволяет понять физическую и математическую сущности моделируемых явлений и обосновать оптимальные подходы к проектированию самых различных изделий.

Реальная польза от моделирования может быть получена при выполнении двух главных условий:

- модель должна быть адекватной оригиналу в том смысле, что должна с достаточной точностью отображать интересующие исследователя характеристики оригинала;
- модель должна устранять проблемы, связанные с физическим измерением каких-то сигналов или характеристик оригинала.

1.1.2. Основные виды моделей

В зависимости от способа реализации все модели можно разделить на два обширных класса.

Физические модели. Они предполагают, как правило, реальное воплощение тех физических свойств оригинала, которые интересуют исследователя. Упрощенные физические модели, нередко уменьшенных габаритов, называются *макетами*. Поэтому физическое моделирование часто именуют *макетированием*.

Математические модели. Они представляют собой формализованные описания объекта или системы с помощью некоторого абстрактного языка, например в виде совокупности математических соотношений или схемы алгоритма. Различают следующие виды математического моделирования: вербальные (словесные), графические, табличные, аналитические и алгоритмические. Нередко математические модели оказываются пригодными для описания множества систем и явлений в самых различных областях науки, техники и экономики.

Иногда математическая модель описывается уравнениями, которые явно вытекают из рассмотрения физической сущности моделируемого явления или системы. Примером может служить экспоненциальное выражение для вольтамперной характеристики полупроводникового диода (теория предсказывает именно такой ее вид). Однако чаще описание моделируемых объектов и систем носит чисто формальный характер и базируется на том, что многие явления порой самой различной природы описываются уравнениями (алгебраическими, дифференциальными и иными) одного и того же вида. В этом случае говорят о *формальных* моделях. Например, формальной моделью того же диода служит модель в виде отрезков двух прямых: один задает сопротивление диода в открытом, а другой – в закрытом состоянии.

Если математическая модель служит для имитации поведения какого-либо реального объекта во времени, то она называется *имитационной моделью*. В англоязычной литературе это соответствует термину Simulation Modeling (в смысле симуляции поведения). К уточнению понятия имитационной модели мы еще вернемся. Пока лишь отметим, что именно имитационное моделирование является основным для пакета расширения Simulink системы MATLAB, что прямо видно из названия этого пакета.

Кроме того, явления, системы и их модели могут быть нестационарными и стационарными. *Нестационарные модели* характеризуются зависимостью их параметров от времени. У *стационарных моделей* такой зависимости нет. Естественно, что моделирование нестационарных явлений гораздо сложнее, чем стационарных.

1.1.3. Основные свойства моделей

Модели обладают рядом свойств, от которых зависит успех их применения в практике моделирования. Отметим лишь некоторые из них, наиболее важные.

- *Адекватность* – это степень соответствия модели исследуемому реальному объекту. Она никогда не может быть полной. На практике модель считают адекватной, если она с удовлетворительной точностью позволяет достичь целей исследования.
- *Простота* (*сложность*) также является одной из характеристик модели. Чем большее количество свойств объекта описывает модель, тем более сложной она оказывается. Не всегда чем сложнее модель, тем выше ее адекватность. Надо стремиться найти наиболее простую модель, позволяющую достичь требуемых результатов изучения.
- *Потенциальность* (*предсказательность*) – способность модели дать новые знания об исследуемом объекте, спрогнозировать его поведение или свойства. На основе изучения математических моделей, описывающих движение планет Солнечной системы с учетом закона всемирного тяготения, теоретически были предсказаны существование и орбиты планет Нептун и Плутон.

Есть и другие свойства моделей, но они не столь важны, как отмеченные.

1.1.4. Цели моделирования

Существует множество конкретных целей моделирования. Отметим две цели обобщающего значения:

- изучение механизма явлений (познавательная цель);
- управление объектами и системами с целью выработки по модели оптимальных управляемых воздействий и характеристик системы.

В обоих случаях модель создается для определения и прогноза интересующих нас характеристик или сигналов объекта.

1.2. Источники воздействий и сигналы

1.2.1. Понятие о сигналах

Чаще всего целью моделирования является изучение реакции системы или устройства на некоторые воздействия, в качестве которых нередко используются *сигналы*. Иногда их называют стандартными, или *тестовыми*, *сигналами*.

Слово «сигнал» происходит от латинского слова «сигнум» – знак. Итак, сигналы – это знаки (символы), о назначении которых мы заранее условились. Такова информационная трактовка сигнала. В физико-математическом представлении под сигналом можно подразумевать функциональную зависимость некоторого параметра (например, напряжения, тока, усилия, расстояния и т. д.) от другого параметра (например, времени, интенсивности света и т. д.). Однако отождествлять сигнал просто с функцией не совсем верно. Сигналы правильно рассматривать как носители информации той или иной физической природы. Это могут быть напряжения и токи (электрические сигналы), колебания воздуха при звуках (звуковые сигналы), электромагнитные волны и свет (оптические сигналы) и т. д.

Сигналы можно рассматривать также как форму, в которую облечена передаваемая, хранимая или перерабатываемая информация. Сигналы могут быть преобразованы из одного вида в другой вид (например, электрические сигналы можно преобразовать в оптические, и наоборот) при сохранении имеющейся в сигнале информации. Сигналы могут быть *стационарными*, если их параметры неизменны в ходе моделирования, или *нестационарными*, если они меняются – чаще всего во времени.

В Simulink принято говорить о воздействиях любой физической природы. В пакете есть обширный набор источников воздействий (сигналов) и средств для простого и быстрого изменения их параметров.

1.2.2. Синусоидальный сигнал

С помощью источников воздействия можно оценивать поведение различных устройств и систем. К примеру, важнейшие характеристики линейных усилителей рассматриваются как его реакция на гармонический (синусоидальный) сигнал:

$$u(t) = U_m \cdot \sin(\omega \cdot t + \varphi) = U_m \cdot \sin(2 \cdot \pi \cdot f \cdot t + \varphi),$$

где U_m – амплитуда сигнала, $\omega = 2\pi f$ – круговая частота (в рад/с), f – частота (в герцах), φ – фаза (в долях периода $T = 1/f$ или градусах). Синусоидальный сигнал является периодической функцией времени t, что соответствует равенству $u(t) = u(t \pm k \cdot T)$, где k – целое число. Синусоидальный сигнал стационарен – это означает, что его параметры (амплитуда, частота и фаза) не меняются во времени. Такой сигнал определен в интервале времени от $-\infty$ до $+\infty$, то есть по существу он является теоретической абстракцией (достаточно отметить, что энергия подобного сигнала равна бесконечности). Естественно, что на практике сигнал такого вида рассматривается в конечном интервале времени.

1.2.3. Дельта-функция Дирака и функция Хевисайда

Целью моделирования импульсных систем и устройств часто является оценка их влияния на импульсные сигналы. В теоретическом аспекте особый интерес представляют два импульсных сигнала – единичный импульс и единичный перепад.

Единичный импульс, или δ-*функция* (дельта-функция Дирака), определяется соотношениями:

$$\delta(t)=\begin{cases}0, & t\neq 0\\ \infty, & t=0\end{cases} \quad \text{и} \quad \int_{-\infty}^{\infty}\delta(t)dt=1.$$

Физически этот сигнал не реализуем, но теоретически реакция на него линейной системы определяет ее *импульсную характеристику*.

Единичный скачок (функция Хевисайда) определяется выражением:

$$\sigma(t)=\begin{cases}0, & t<0\\ \dfrac{1}{2}, & t=0\\ 1, & t>0\end{cases}.$$

Единичный перепад легко реализуется физически, а реакция системы на него есть ее *переходная характеристика*. Иногда применяются несколько отличные выражения для единичного перепада, например считают его значение равным 1 при $t \geq 0$.

Существует множество импульсных сигналов самой различной формы. Например, прямоугольный импульс единичной амплитуды и заданной длительности t_u нетрудно получить, сложив с единичным перепадом другой перепад (с 0 на −1) с задержкой t_u. Импульсные сигналы различной формы (прямоугольной, треугольной, пилообразной и прочей) создаются источниками сигналов, блоки которых есть в программах математического моделирования, например Simulink.

Реакция системы или устройства на входные сигналы, в свою очередь, может быть представлена *выходными сигналами*. Таким образом, вполне правомерно считать, что системы моделирования относятся к информационным системам, задача которых заключается в обработке некоторого множества входных сигналов с целью получения множества выходных сигналов. Последние могут использоваться для управления объектами и их контроля, либо просто для получения информации о закономерностях работы тех или иных систем и устройств.

1.3. Технология моделирования

1.3.1. Комплексное моделирование

Степень реализации перечисленных принципов каждой конкретной модели может быть различной. Это зависит не только от желания исследователя, но и от соблюдения им технологий моделирования. А любая технология подразумевает определенную последовательность действий.

В настоящее время самой распространенной *технологией моделирования* является *комплексное моделирование*, под которым понимается математическое моделирование с использованием средств вычислительной техники. Соответствующие технологии комплексного моделирования представляют выполнение следующих действий:

- определение цели моделирования;
- разработка концептуальной модели;
- формализация модели;
- программная реализация модели;
- планирование модельных экспериментов;
- реализация плана эксперимента;
- анализ и интерпретация результатов моделирования.

Результаты комплексного моделирования используются как основа для дальнейших исследований и разработок, в том числе дорогостоящих натурных испытаний.

1.3.2. Основные методы решения задач моделирования

На этапе программной реализации модели и реализации плана экспериментов необходим выбор методов решения задач моделирования. При этом используются три основные группы методов:

- *графические* – оценочные приближенные методы, основанные на построении и анализе графиков;
- *аналитические* – решения, строго полученные в виде аналитических выражений (пригодны для узкого круга задач);
- *численные* – основной инструмент для решения сложных математических задач, основанный на применении различных численных методов.

Simulink – система моделирования, основанная на применении численных методов, которые реализованы матричной системой MATLAB.

1.3.3. Погрешности моделирования

Аналитическое решение удается получить редко и чаще всего лишь при упрощенной формулировке задачи моделирования в линейном приближении. Основным средством моделирования является алгоритмический подход, реализующий вычислительный эксперимент на ЭВМ или в наши дни на *персональном компьютере* (ПК). Получаемое на ЭВМ решение почти всегда содержит некоторую *погрешность*. Абсолютная погрешность

$$\varepsilon = x - x_u$$

есть разность между приближенным x и точным, или идеальным, x_u значениями результата, а относительная погрешность определяется как

$$\Delta = \varepsilon / x_u.$$

Наличие погрешности решения обусловлено рядом причин. Перечислим основные источники погрешности.

1. Математическая модель является лишь приближенным описанием реального процесса (погрешность модели).
2. Исходные данные, как правило, содержат погрешности, поскольку являются результатами приближенных экспериментов (измерений), или решениями вспомогательных задач (погрешность данных).
3. Применяемые для решения задачи методы в большинстве случаев являются приближенными (погрешность метода).
4. При вводе исходных данных в ЭВМ, выполнении операций производятся округления (вычислительная погрешность).

Погрешности 1 и 2 – не устранимые на данном этапе решения, для их уменьшения приходится возвращаться вновь к построению математической, а и иногда и концептуальной модели, проводить дополнительное экспериментальное уточнение условий задачи.

1.3.4. Оценка обусловленности вычислительной задачи

Оценка обусловленности вычислительной задачи – еще одно обязательное требование при выборе метода решения и построении математической модели.

Пусть вычислительная задача корректна. Теоретически устойчивость задачи означает, что ее решение может быть найдено со сколь угодно малой погрешностью, если только гарантировать достаточно малую погрешность входных данных. Однако на практике их точность ограничена (и величиной гораздо большей, чем $\varepsilon_м = 2^{-P+1}$ – машинная точность, p – порядок, округление производится усечением).

Как влияют малые, но конечные погрешности входных данных на решение? Насколько сильно они искажают результат? Ответ на это дает понятие *обусловленности* задачи, то есть чувствительности решения вычислительной задачи к малым погрешностям входных данных.

Задачу называют *хорошо обусловленной*, если малым погрешностям входных данных отвечают малые погрешности решения, и *плохо обусловленной*, если возможны сильные изменения решения. Часто возможно ввести количественную оценку степени обусловленности – *число обусловленности*: его можно интерпретировать как коэффициент возможного возрастания погрешности в решении по отношению к вызвавшей их погрешности входных данных. Если установлено неравенство между этими погрешностями, то можно пользоваться следующими выражениями:

$$\Delta(y^*) \geq \nu_\Delta \cdot \Delta(x^*) \quad \text{и} \quad \delta(y^*) \leq \nu_\delta \cdot \delta(x^*),$$

где ν_Δ – абсолютное число обусловленности, ν_δ – относительное число обусловленности. Для плохо обусловленных задач $\nu_\delta >> 1$, неустойчивость соответствует $\nu_\delta = \infty$.

При каких значениях ν_δ можно считать задачу плохо обусловленной? Это зависит от требований к точности решения и от уровня обеспечиваемой точности исходных данных.

Если требуется найти решение с точностью 0,1%, а входная информация задается с точностью в 0,02%, то при $\nu_\delta = 10$ уже будет плохая обусловленность.

Однако если исходные данные задаются с $\delta(x^*) \leq 0{,}0001\%$, то при $\nu_\delta = 10^3$ – задача хорошо обусловлена ($\delta(y^*) = 0{,}1\%$).

1.3.5. Вычислительные методы в моделировании

Моделирование реализуется различными вычислительными методами. Вычислительные методы преобразуются к виду, удобному для программной реализации. Можно выделить следующие классы численных методов:

- *метод эквивалентных преобразований* – исходную задачу заменяют другой, имеющей то же решение: нахождение корня нелинейного уравнения $f(x) = 0$ сводят к поиску точек глобального минимума $\Phi(x) = (f(x))^2$;
- *методы аппроксимации* – заменяют исходную задачу другой, решение которой близко к решению исходной задачи;
- *методы конечно-разностные*, основанные на замене производных конечными разностями, например $f'(x) \approx \dfrac{f(x+h) - f(x)}{h}$;
- *прямые* (*точные*) *методы* – решение может быть получено за конечное число элементарных операций (арифметические и извлечение корня). Многие прямые методы не годятся к применению в ЭВМ из-за чувствительности к ошибкам округления;
- *итерационные методы* – методы последовательных приближений к решению задачи. Задается начальное приближение решения, строится итерационная последовательность приближений к решению. Если эта последовательность сходится к решению, то говорят, что итерационный процесс сходится. Множество начальных приближений, для которых метод сходится, называются областью сходимости метода;
- *методы статистических испытаний* (*Монте-Карло*) – основаны на моделировании случайных величин и построении статистических оценок решений задач (для моделирования больших систем). Для реализации этих методов используются генераторы случайных чисел.

Численные методы группируются вокруг типичных математических задач: задач анализа, алгебры, оптимизации, решения дифференциальных и интегральных уравнений, обратных задач (синтез). Этот этап решения заканчивается выбором и обоснованием конкретных численных методов решения, разработкой алгоритма, которые могут быть программно реализованы средствами компьютерной техники.

1.3.6. Контроль правильности модели

Для контроля правильности полученной модели может использоваться ряд приемов:

- *анализ размерности* – величины в левой и правой частях выражения, отдельные слагаемые в каждой из частей должны иметь одинаковую размерность;

- *проверка порядков и характеров зависимостей* – параметры и переменные, которые в данной задаче выражены величинами большего порядка малости, могут быть исключены из рассмотрения как несущественные, что часто позволяет значительно упростить модель и ее анализ. Характер изменения значений моделируемых величин должен соответствовать их реальному смыслу, не противоречить наблюдаемым данным;
- *исследование предельных случаев* – результаты моделирования при крайних значениях параметров модели, равных, как правило, нулю или бесконечности, не должны противоречить смыслу (например, энергия реальной физической системы не может оказаться бесконечно большой, время протекания процесса – отрицательным и т. п.). Модель в этом случае существенно упрощается и легче для понимания;
- *проверка замкнутости и корректности математической задачи* – система математических соотношений должна иметь единственное решение.

Задача называется *корректной*, если она удовлетворяет трем требованиям:

- ее решение существует при любых допустимых входных данных;
- это решение единственно (однозначно определено);
- решение непрерывно зависит от данных задачи – устойчиво по отношению к малым возмущениям входных данных.

Решение вычислительной задачи называется *устойчивым* по входным данным X, если оно зависит от входных данных непрерывным образом; то есть для любого $\varepsilon > 0$ существует $\delta = \delta(\varepsilon) > 0$ такое, что всяким исходным данным x^*, удовлетворяющим условию $\Delta(x^*) < \delta$, отвечает приближенное решение y^*, для которого $\Delta(y^*) < \varepsilon$.

Далеко не все встречающиеся на практике задачи являются корректными. К ним, например, нельзя отнести обратные задачи геофизики, астрофизики, спектрографии, распознавания образов, синтез и многие другие важные прикладные проблемы. Свойство корректности задачи имеет большое значение для выбора метода решения. К некорректным задачам неприменимы обычные численные методы вычислительной математики. Строгий анализ корректности во многих случаях математически сложен, и ограничиваются проверкой соответствия количества неизвестных и связывающих их уравнений в модели.

1.4. Моделирование линейных динамических объектов и систем

1.4.1. Идентификация динамических объектов

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

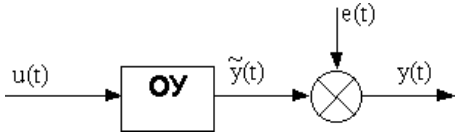


Рис. 1.1. Общее представление идентифицируемого объекта О

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

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

при этом наблюдаемый выход объекта может быть описан соотношением

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

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

1.4.2. О моделировании линейных систем

Многие системы относятся к классу *линейных систем*. У таких систем временная зависимость отклика системы на входные воздействия не зависит от их уровней. Для линейных систем действует *принцип суперпозиции* – отклик такой системы на сумму ряда воздействий равен сумме откликов на каждое воздействие. Частным примером его эффективного применения является разложение сложного сигнала в ряд Фурье, анализ реакции системы на каждую гармонику и вычисление реакции системы как суммы воздействий на каждую гармонику. В этом состоит суть *спектрального метода моделирования* линейных систем.

Линейные системы строятся из линейных блоков, таких как усилители без ограничения сигналов, идеальные интеграторы и т. д. При этом возникает задача определения параметров таких блоков (объектов).

1.4.3. Простая линейная модель RC-цепи

Иногда задачу определения математической зависимости можно решить чисто аналитическим путем. Примером может служить простая RC-цепь, показанная на рис. 1.2.

Исходя из известных законов электротехники, для выходного сигнала этой цепи можно записать следующее общеизвестное выражение:

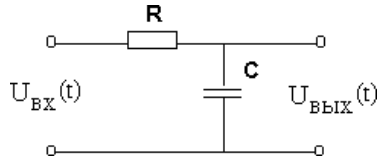


Рис. 1.2. Простая RC-цепь

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

где Uвх$(t) = u(t)$, Uвых$(t) = y(t)$.

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

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

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

Последнее уравнение позволяет после проведения эксперимента, заключающегося в фиксации входного и выходного сигналов на каком-то интервале времени, провести обработку экспериментальных данных и каким-либо методом (например, методом наименьших квадратов) найти оценку вектора параметров β.

1.4.4. Передаточная функция

При моделировании линейных систем часто используется операторный метод. Особенно широко он применяется в электро- и радиотехнике. Например, в линейных цепях замена применения операторного метода позволяет свести дифференциальные уравнения к алгебраическим уравнениям и резко упростить анализ цепей.

Одним из главных понятий линейных систем, анализируемых операторным методом, является *передаточная характеристика* – отношение преобразований Лапласа выходного и входного сигналов. Она записывается следующим образом:

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

где $L\{\cdot\}$ – символ преобразования Лапласа, p (или s) – комплексная переменная, именуемая оператором Лапласа. В общем случае эта характеристика записывается как отношение двух полиномов – числителя и знаменателя. Выражение $W(p)$ дано, как принято в нашей литературе. В западных СКМ используется оператор Лапласа s, коэффициенты полинома числителя обозначены буквой a, а знаменателя – буквой b.

Корни полинома числителя создают нули передаточной характеристики, а корни полинома знаменателя – полюсы. Многие СКМ и программы моделирования позволяют строить графики передаточных характеристик и вычислять их нули и полюсы. Изучая положение нулей и полюсов на комплексной плоскости, можно судить о свойствах системы.

1.4.5. Импульсная характеристика (ИХ) w(t)

Под *импульсной характеристикой* (ИХ) понимается реакция предварительно невозмущенного объекта (то есть объекта с нулевыми начальными условиями) на входной сигнал в виде δ-функции (дельта-функции Дирака, или единичной функции). Импульсная функция является производной от переходной функции, описанной ниже. Как уже отмечалось, дельта-функция Дирака практически не реализуема, тем не менее импульсная характеристика линейных систем и устройств является вполне реальной характеристикой, широко используемой на практике. Многие системы моделирования имеют средства для ее построения.

1.4.6. Переходная характеристика, или функция h(t)

Переходная характеристика (ПХ) – это реакция предварительно невозмущенного объекта на входной сигнал в виде *единичного скачка*. Эта характеристика определяется как

$$h(t) = \int_{-\infty}^{t} w(\tau)d\tau.$$

В практике математического моделирования используются следующие соотношения:

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

В частности, они позволяют переходить от определения импульсных характеристик к определению переходных характеристик, и наоборот. Построение переходных характеристик входит в набор средств большинства систем математического моделирования, включая Simulink.

1.4.7. Свертка и интеграл свертки

Любой сигнал может быть представлен в виде так называемой *свертки* самого себя с δ-функцией:

$$u(t) = \int_{-\infty}^{\infty} u(\tau)\delta(t-\tau)d\tau.$$

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

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

или, в операторной форме:

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

Здесь уместно отметить, что иногда импульсную характеристику обозначают как $h(t)$, а переходную – как $g(t)$. Соответственно, входной сигнал $u(t)$ часто обозначают как $x(t)$. К сожалению, это служит причиной путаницы. Поэтому важно понимать разницу между ними и использовать соответствующие выражения по контексту.

1.4.8. Основы спектрального анализа и синтеза

Одним из эффективных методов моделирования сигналов и линейных систем является спектральный метод, основанный на применении преобразований и рядов Фурье. Напомним, что рядом Фурье для интегрируемой на отрезке $[-\pi, \pi]$ периодической функции $y(x)$, удовлетворяющей известным условиям Дирихле, называют следующий ряд:

$$y(x) \approx \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos(kx) + b_k \sin(kx)\right).$$

Коэффициенты Фурье этого ряда находятся по формулам Эйлера–Фурье:

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} y(x)\cos(kx)\,dx; \quad b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} y(x)\sin(kx)dx.$$

Важными сферами применения рядов Фурье являются радиотехнические расчеты. В них периодические сигналы обычно представляют как функции времени $y(t)$ на отрезке $[0, T]$ с периодом $T = 1/f_1$, где f_1 – частота первой гармоники периодического сигнала. В этом случае ряд Фурье после несложных преобразований записывается в тригонометрическом виде:

$$y(t) \approx \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos(2\pi k f_1 t) + b_k \sin(2\pi k f_1 t)\right).$$

Здесь коэффициенты выглядят следующим образом:

$$a_k = \frac{2}{T}\int_0^T y(t)\cos(2\pi k f_1 t)\,dt; \quad b_k = \frac{2}{T}\int_0^T y(t)\sin(2\pi k f_1 t)\,dt.$$

В этом случае коэффициенты a_k и b_k описывают косинусную и синусную составляющие k-й гармоники сигнала с периодом T и частотой повторения $f_1 = 1/T$. Часто используется иная форма ряда Фурье, упрощающая его синтез:

$$y(t) \approx \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(A_k \cos(2\pi k f_1 t) + \varphi_k\right).$$

Здесь A_k – амплитуда k-й гармоники периодического сигнала, φ_k – фаза k-й гармоники. Они вычисляются по формулам:

$$A_k = \sqrt{a_k^2 + b_k^2}; \quad \varphi_k = -\arctan(b_k / a_k).$$

Разложение функции на гармонические составляющие, то есть вычисление коэффициентов Фурье, принято называть *спектральным анализом*. А воссоздание приближения функции рядом Фурье, то есть получение ее тригонометрического представления, называют *спектральным синтезом*.

Гармонику с $k = 1$ называют *основной*, или *первой*, *гармоникой* сигнала. Она задает его частоту повторения f_1. Остальные гармоники называют *высшими*, их частоты равны $f_k = k \cdot f_1$, где $k = 2, 3, \ldots$. Таким образом, спектр периодических сигналов *дискретный* – он содержит набор фиксированных частот f_k, где $k = 1, 2, 3, \ldots$. У непериодических сигналов спектр будет *сплошным*, и вместо амплитуды гармоник он характеризуется *спектральной плотностью* сигнала.

Далее будем рассматривать сигналы как функции времени. Переход от некоторой функции $f(t)$ к параметрам ее ряда Фурье (амплитудам и фазам гармоник) называется *прямым преобразованием Фурье*, а обратный переход – *обратным преобразованием Фурье*. К сожалению, эти переходы связаны с вычислением интегралов, подынтегральные функции в которых быстро осциллируют, что существенно затрудняет вычисление таких интегралов численными методами с заданной точностью и ведет к значительным затратам времени.

Если сигнал представлен в виде вектора дискретных значений, применяется *дискретное* преобразование Фурье (ДПФ), для которого, в свою очередь, существует алгоритм эффективной реализации вычислений, называемый *быстрым* преобразованием Фурье (БПФ, или FFT – Fast Fourier Transform). Функции, реализующие прямое и обратное БПФ, есть в системе MATLAB. Они предоставляют возможность проводить указанные преобразования для данных в виде векторов как с действительными, так и с комплексными элементами.

Выполнение БПФ производится для данных, представленных действительными числами – значениями исходного вектора ν. Он должен иметь $2m$ составляющих, где m – целое число. Элементы вектора, возвращаемого функцией прямого БПФ, соответствуют формуле

$$C_j = \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} \nu_k e^{2\pi i (j/n)k}.$$

Здесь n – число элементов вектора ν, i – мнимая единица, k – индекс суммирования (от 0 до $n-1$) и j – номер гармоники (от 0 до $n/2$). Эти элементы вектора соответствуют следующим частотам:

$$f_j = \frac{j}{n} \cdot f_s.$$

Здесь f_s – частота квантования сигнала, который подвергается БПФ. Элементы вектора, возвращаемого этой функцией, – в общем случае комплексные числа, даже если сигнал представлен вещественными отсчетами.

Функция обратного БПФ реализует обратное (инверсное) преобразование Фурье для вектора v с комплексными элементами. Это преобразование реализуется по формуле

$$d_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} w_k e^{-2\pi i (j/n)k}.$$

Рассмотренные выше функции основаны на обычных формулах преобразований Фурье. Однако существуют и альтернативные формы такого преобразования, две из которых показаны ниже:

$$F(\nu) = \frac{1}{n} \sum_{\tau=1}^{n} f(\tau) e^{-2\pi i \tau(\nu/n)} \quad \text{и} \quad f(\tau) = \sum_{\nu=1}^{n} F(\nu) e^{2\pi i \nu(\tau/n)}.$$

Вместо множителя $1/\sqrt{n}$ перед обоими выражениями перед первым выражением стоит множитель $1/n$, а перед вторым – 1. Знак «минус» перед показателем степени имеется только в первой формуле (его нет во второй).

В общем случае, когда сигнал может быть и непериодическим, прямое преобразование Фурье позволяет получить в аналитическом виде функцию частоты $F(\omega)$ от временной функции $f(t)$. Оно реализуется формулой

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i \cdot \omega \cdot t} dt.$$

Соответственно, обратное преобразование Фурье задается следующим образом:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i \cdot \omega \cdot t} d\omega.$$

Эта формула позволяет по функции $F(\omega)$ найти в аналитическом виде функцию $f(t)$. Интересно, что если подвергнуть преобразованию Фурье свертку для линейной системы, то можно получить подтверждение вполне очевидного вывода – спектр свертки равен произведению спектров. Это еще один вывод из линейности Фурье-преобразований.

1.4.9. Частотные характеристики

Частотные характеристики объекта определяются его комплексным коэффициентом передачи $W(j\omega) = W(p)|p = j\omega$, который является Фурье-преобразованием ИХ. В радиотехнике и электронике вместо коэффициента передачи пользуются понятием комплексного коэффициента усиления $K(j\omega) = \dot{K}(\omega)$ [34].

Модуль комплексного коэффициента передачи $|W(j\omega)| = A(\omega)$ представляет собой, как известно, *амплитудно-частотную характеристику* (АЧХ) объекта с передаточной функцией $W(p)$, а аргумент $\arg(W(j\omega)) = \varphi(\omega)$ – *фазо-частотную характеристику* (ФЧХ). В электронике АЧХ и ФЧХ являются важнейшими параметрами линейных усилителей, выполненных на электронных лампах, транзисторах или интегральных микросхемах.

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

В теории управления, да и в радиоэлектронике, часто используется *логарифмическая амплитудно-частотная характеристика* (ЛАЧХ), определяемая выражением $20 \lg |W(j\omega)|$. При этом ось частот также строится в логарифмическом масштабе.

1.5. Моделирование нелинейных объектов и систем

1.5.1. Дифференциальное уравнение

Дифференциальным уравнениям принадлежит особая роль в математическом моделировании [71–80]. Прежде всего это связано с тем, что они позволяют моделировать не только линейные, но и нелинейные устройства (объекты) и системы. Для последних получение аналитических решений в общем случае невозможно.

Наиболее универсальная модель объекта имеет вид дифференциального уравнения

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

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

Дифференциальные уравнения, описывающие простые линейные (очень редко и нелинейные) системы, имеют аналитические решения. Иногда их можно найти в аналитическом виде, например используя функции символьных вычислений, присущие многим системам компьютерной математики. Но, увы, чаще всего полученные решения оказываются настолько громоздкими, что практическая польза от них становится весьма сомнительной или попросту отсутствует. В таких условиях решение даже систем линейных уравнений вполне оправдано численными методами, что и реализовано в пакете расширения Simulink.

В случае нелинейных объектов и систем параметры дифференциальных уравнений перестают быть постоянными и зависят от уровней переменных. При чис-

ленных методах решения дифференциальных уравнений задание таких зависимостей не представляет трудностей, а потому дифференциальные уравнения (или алгебраически-дифференциальные уравнения) и составляют основу математического моделирования нелинейных устройств и систем. В Simulink имеется обширный набор решателей дифференциальных уравнений.

1.5.2. Модель для переменных состояния

В данной книге под *имитационной моделью* подразумевается логико-математическое описание системы, которое может быть исследовано с помощью цифровой ЭВМ в ее современном виде – в виде персонального компьютера (ПК). Ключевым моментом в этом случае являются выделение и описание *состояний* системы. Каждое состояние характеризуется набором значений некоторых переменных, называемых переменными состояния.

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

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

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

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

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

где $\boldsymbol{I}$ – единичная матрица.

Отметим, что все приведенные модели являются эквивалентными, то есть, зная любую из них, можно получить все остальные. Модель переменных состояния широко используется в системах блочного имитационного моделирования, таких как Simulink.

1.6. Моделирование дискретных систем

1.6.1. Дискретные модели и Z-преобразования

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

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

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

Это уравнение выполняет ту же роль, что дифференциальное уравнение при описании непрерывных объектов. Связь между сигналами может быть отражена также через дискретную свертку

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

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

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

где $z = e^{pT}$, через дискретную передаточную функцию

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

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

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

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

Отметим далее, что на практике в большинстве случаев измерение непрерывных сигналов производится в дискретные моменты времени, что представляет определенное удобство при последующей обработке данных на ПК. При этом существуют различные способы перехода от непрерывных моделей к дискретным моделям:

- с применением Z-преобразования со следующей цепочкой переходов:
  $W(p) \to L^{-1}\{W(p)\} = w(t) \to w(kT) = w_k \to W(z) = Z\{w_k\}$;
- с заменой производных в дифференциальном уравнении, описывающем непрерывный объект, разностями:
  $\frac{dy(t)}{dt} \approx \frac{y_k - y_{k-1}}{T}$; $\frac{d^2 y(t)}{dt^2} \approx \frac{y_k - 2y_{k-1} + y_{k-2}}{T^2}$
  и т. д. (данный подход дает приемлемую точность только при малых T);
- с заменой $p = \frac{2}{T} \cdot \frac{z-1}{z+1}$ (приближенный способ, предложенный А. Тастиным и называемый билинейным преобразованием), то есть
  $W(p)\big|_{p=\frac{2}{T}\cdot\frac{z-1}{z+1}} \to W(z)$.

1.6.2. Дискретные модели переменных состояния

Для дискретных объектов также может быть использовано описание через переменные состояния

$\boldsymbol{X}_k = \boldsymbol{A}\boldsymbol{X}_{k-1} + \boldsymbol{B}u_{k-1}$,
$y_k = \boldsymbol{C}\boldsymbol{X}_k + \boldsymbol{D}u_k$,

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

Отметим, что множитель $z^{-1} = e^{-pT}$ представляет собой *оператор задержки*, то есть $z^{-1}u_k = u_{k-1}$, $z^{-2}u_k = u_{k-2}$ и т. д.

1.6.3. Некоторые понятия статистического моделирования

Большинство систем математического моделирования реализуют в той или иной мере возможности статистического моделирования. Познакомимся кратко с основными понятиями, относящимися к этому (более подробные сведения можно найти в многочисленной специальной литературе по статистике и статистическим методам вычислений).

Создание некоторой системы условий называют *испытанием*, или *экспериментом*. Если до осуществления эксперимента его результаты нельзя точно предсказать, то эксперимент называют *вероятностным*, *случайным* или *стохастическим*. В ходе эксперимента происходят факты или события A, наступление которых можно наблюдать. Изучением законов, которым подчиняются случайные события, занимается *теория вероятности*.

События могут быть достоверными Ω, невозможными ∅ и случайными. В последнем случае события могут наступать или не наступать. Пара событий может быть несовместной, если наступление одного события исключает другое (например, падение монеты на ту или иную сторону). События могут быть взаимно противоположными, если они несовместны и одно из них наступает. Возможны объединения (суммы) и пересечения событий.

Случайные события характеризуются *вероятностью события* $P(A)$, которую оценивают числом от 0 (событие не наступает) до 1 (при 1 событие непременно наступит). Если число равновозможных элементарных исходов некоторого эксперимента равно n, а событию A благоприятствует m исходов, то *классическая вероятность* события A будет $P(A) = m/n$. Пусть на тарелке лежат 10 белых и $m = 5$ красных черешен. Значит, $n = 10 + 5 = 15$. Какова вероятность, что мы возьмем наугад красную черешню? Она равна $P(A) = m/n = 5/15 = 1/3$.

Классическое определение вероятности неприемлемо, если события не являются равновозможными. Например, игральный кубик со скошенными некоторыми гранями не имеет равновозможных вариантов выпадения. В таких случаях пользуются *статистической вероятностью* событий. Пусть при n экспериментов событие A наступило m раз. Это число называют абсолютной частотой события A, а $P^*(A) = m/n$ – относительной частотой события. Вероятностью события A называют число $P(A)$, около которого группируются значения относительной частоты события A при большом числе экспериментов (испытаний).

Математическая статистика – это наука о методах систематизации и использования статистических данных для получения научных и практических выводов. Она решает множество полезных задач. Из них мы ограничимся только теми, которые изначально заложены в пакет расширения Simulink. В основном это задачи, которые могут решаться с помощью генераторов случайных чисел. При этом мы будем рассматривать некоторую совокупность данных, называемую генеральной совокупностью, а также выборки данных из нее, именуемые выборочными совокупностями. Как правило, данные мы будем представлять в виде *вариационного ряда*, при котором они используются в порядке их возрастания.

1.6.4. Дискретные модели, учитывающие шум наблюдения

Часто данные наблюдения содержат случайную компоненту – шум. Обозначив моменты дискретного времени тем же символом t, что и непрерывное время (в данном случае $t = 0, 1, 2, ...$), приведем несколько распространенных моделей дискретных объектов для временной области, учитывающих действие шума наблюдения.

- Модель авторегрессии AR (AutoRegressive) – считается самым простым описанием:
  $A(z)\,y(t) = e(t)$,
  где $A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_{na} z^{-na}$;
- ARX-модель (AutoRegressive with eXternal input) – более сложная:
  $A(z)\,y(t) = B(z)\,u(t) + e(t)$, или, в развернутом виде,
  $y(t) + a_1 y(t-1) + ... + a_{na} y(t-n) = b_1 u(t) + b_2 u(t-1) + ... + b_{nb} u(t-m) + e(t)$.
  Здесь и ниже $e(t)$ – дискретный белый шум, $B(z) = b_1 + b_2 z^{-1} + ... + b_{nb} z^{-nb+1}$;
- ARMAX-модель (AutoRegressive-Moving Average with eXternal input – модель авторегрессии скользящего среднего):
  $A(z)\,y(t) = B(z)\,u(t - nk) + C(z)\,e(t)$,
  где n_k – величина задержки (запаздывания), $C(z) = 1 + c_1 z^{-1} + c_2 z^{-2} + ... + c_{nc} z^{-nc}$;
- модель «вход-выход» (в англоязычных источниках такая модель называется «Output-Error», то есть «выход-ошибка», сокращенно OE):
  $$y(t) = \frac{B(z)}{F(z)} u(t - nk) + e(t),$$
  где $F(z) = 1 + f_1 z^{-1} + f_2 z^{-2} + ... + f_{nf} z^{-nf}$;
- так называемая модель Бокса–Дженкинса (BJ):
  $$y(t) = \frac{B(z)}{F(z)} u(t - nk) + \frac{C(z)}{D(z)} e(t);$$
- (полиномы $B(z)$, $F(z)$, $C(z)$ определены ранее, а $D(z) = 1 + d_1 z^{-1} + d_2 z^{-2} + ... + d_{nd} z^{-nd}$).

Данные модели можно рассматривать как частные случаи обобщенной параметрической линейной структуры

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

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

Модель для переменных состояния (State space):

$$x(t+1) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t),$$
$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t) + v(t),$$

где $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ – матрицы соответствующих размеров, $v(t)$ – коррелированный шум наблюдений.

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

$$x(t+1) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) + \boldsymbol{K}e(t),$$
$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t) + e(t),$$

где $\boldsymbol{K}$ – некоторая матрица (вектор-столбец), $e(t)$ – дискретный белый шум (скаляр). Следует быть внимательными при анализе сигналов, содержащих шумовую компоненту, поскольку под $e(t)$ часто обозначают полный сигнал.

Урок 2

Simulink – пакет визуального математического моделирования

Этот урок является введением в основной пакет расширения системы MATLAB – Simulink. Его надо обязательно проработать, прежде чем приступать к практической работе с этим пакетом. В данном уроке отражены некоторые отличия версий пакета Simulink 5, 6 и 7. Хотя они не принципиальны, познакомиться с ними полезно.

2.1. Основные возможности пакета Simulink

2.1.1. Назначение пакета

Как уже отмечалось, пакет расширения Simulink системы MATLAB является ядром интерактивного программного комплекса, предназначенного для *математического моделирования* линейных 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 блоков), пользователь с помощью мыши переносит нужные блоки с палитр на рабочий стол пакета Simulink и соединяет линиями входы и выходы блоков. Таким образом, создается *диаграмма* (блок-схема) системы или устройства, то есть модель.

S-модель фактически является программой, которую можно просмотреть с помощью тестового редактора или (в MATLAB 7.*/R2006*/R2007*) с помощью редактора файлов системы MATLAB. Файлы модели имеют расширение .mdl. Однако следует отметить, что эти файлы очень громоздки и даже для довольно простых моделей могут содержать тысячи строк программного кода. Это типичное свойство визуально-ориентированных систем программирования. В связи с этим в дальнейшем работа моделей будет рассматриваться только на уровне диаграмм и блоков, без рассмотрения программных кодов моделей. Разумеется, опытный пользователь-программист может работать с программными кодами S-моделей и модернизировать их.

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» как более благозвучный и привычный для нашей литературы.

Средства визуализации результатов моделирования в пакете Simulink настолько наглядны, что порой создается ощущение, что созданная диаграмма (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 воздействий (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, заданных параметрически и 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 подобных средств необходимы высокоскоростные решающие устройства. Интеграция одной из самых быстрых матричных математических систем – MATLAB с пакетом Simulink – открыла 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 Simulink 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. Необходимо отметить также возможность моделирования устройств и систем в реальном масштабе времени.

Как программное средство Simulink – типичный представитель визуально-ориентированных языков программирования. На всех этапах работы, особенно при подготовке моделей систем, пользователь практически не имеет дела с обычным программированием. Программа в кодах 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.

Разработчики системы MATLAB + Simulink отказались от конкуренции с разработчиками программ узкого назначения, например схемотехнического. Они сосредоточили свое внимание на решении куда более важной и сложной задачи – моделировании блочных динамических систем и устройств произвольного назначения. Это физические и химические системы и устройства, электротехнические устройства (и даже целые энергетические системы), механические системы и устройства и т. д. и т. п. Для этого пришлось существенно расширить библиотеки компонентов таких систем и устройств, с одной стороны, а с другой – применить укрупненные модели ряда компонентов. Именно благодаря этому возможно моделирование сложных систем и устройств.

Одной из самых сложных проблем в реализации математического моделирования в среде системы MATLAB стала подготовка модели моделируемой системы или устройства. Модель обычно представляется в форме графического, табличного или таблично-топологического описания. При этом необходимо предусмотреть организацию связей между компонентами и установку их, подчас многих, параметров. После этого надо запустить созданную модель на исполнение, то есть задать решение автоматически составленной системы уравнений состояния и вывод результатов решения. Это также представляет собой достаточно сложную проблему.

Все эти проблемы блестяще решены введением в MATLAB второй важной части системы – расширения Simulink. Это расширение реализует по существу визуально-ориентированное программирование задач автоматического составления графической модели системы или устройства, составления и решения ее уравнений состояния и наглядного представления результатов моделирования.

В состав системы MATLAB 6.5 SP1 входил пакет моделирования динамических систем – Simulink 5.1 (R13SP1), выпущенный в конце августа 2003 г. и широко используемый и поныне. Но в систему MATLAB 7 вошла уже расширенная реализация этого пакета Simulink 6.*, которая незначительно модернизировалась по мере усовершенствования базовой системы MATLAB. Так, в версии MATLAB R2007a, детально описанной в данной книге, используется версия Simulink 6.6. А в недавно выпущенной на рынок MATLAB R2007b использован пакет расширения Simulink 7. Его отличия от Simulink 6.* носят скорее символический, чем принципиальный характер (см. ниже).

Далее все версии (Simulink 5.* , Simulink 6.* и Simulink 7) и их модификации описываются под именем Simulink параллельно с выделением, где это необходимо, новых возможностей Simulink 6.*/7. Пакет расширения Simulink 5 описан в [16, 33], и его описание в этой книге дано в минимальном объеме.

2.1.2. Общие возможности Simulink

Уже версия пакета Simulink 3.1 обладала следующими важными возможностями, сохранившими свое значение и в более поздних версиях:

- интегрированный браузер моделей;
- 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;
- 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.

2.1.3. Дополнительные возможности Simulink

В 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 система Simulink 4 подверглась не «косметической», а самой серьезной переработке, выдвигающей эту систему на новый уровень развития и применения.

2.1.4. Новые возможности Simulink 5.*

Реализации Simulink 5 пополнились следующими новыми возможностями:

- введена цветовая индикация ошибочных циклов;
- улучшено исполнение ветвлений входа;
- введен новый блок конструирования источников (сигналов);
- введено средство обзора моделей;
- расширены возможности браузера диагностики;
- расширены возможности редактора масок;
- введен редактор включения специальных символов и греческих букв;
- расширены возможности модели дискретизатора.

Эти возможности трудно назвать заметными, за исключением, пожалуй, блока задания источников (сигналов). Однако в MATLAB 6.5 Service Pack 1 Simulink 5 был существенно переработан и практически превратился в Simulink 6.

2.1.5. Новые возможности Simulink 6.*

Simulink 6 – это существенно обновленная платформа для эмуляции и модельного блочного проектирования динамических систем. Она обеспечивает интерактивную графическую среду и настраиваемый набор библиотек блоков, которые

позволяют с высокой точностью проектировать, моделировать, реализовывать и тестировать системы управления, обработки сигналов, связи и т. п. Версия Simulink 6 улучшает производительность, скорость реакции, точность моделирования и эффективность протекания процессов при моделировании больших систем. Simulink 6 содержит новые средства для повышения скорости работы моделей систем управления, связи и обработки сигналов:

- компонентное моделирование больших систем;
- возможность сегментирования модели на несколько файлов, каждый из которых представляет собой отдельную модель;
- возможность моделировать, тестировать и реализовывать каждый компонент отдельно, еще до его вставки в общую модель системы;
- улучшенная интеграция моделей с существующими системами управления файлами и контроля версий;
- инкрементальная загрузка моделей и генерация кода;
- увеличена скорость обновления диаграмм и моделирования для больших моделей;
- созданы рабочие пространства моделей (Model Workspaces), обеспечивающие отдельные области памяти для хранения параметров и переменных каждой модели;
- улучшена поддержка шин для задания интерфейсов, поддержки операций над сигналами шины и описания шин как структур при генерации кода;
- реализована интеграция Simulink и Stateflow;
- унифицированный браузер моделей (Model Explorer) позволяет просматривать, создавать, конфигурировать все сигналы, параметры и свойства моделей;
- объединены и унифицированы настройки параметров моделирования и генерации кода;
- введена поддержка создания и сохранения конфигураций параметров моделирования и генерации кода;
- введена возможность управления данными и их визуализацией;
- добавлены новые объекты данных для задания структур, шин и типов данных;
- введены возможности протоколирования данных и добавления контрольных точек без добавления блоков к модели;
- средство Signal & Scope Manager позволяет подключать к модели источники и приемники сигналов без добавления блоков;
- поддержка языка MATLAB;
- генерация C-кода и реализация приложений на основе внедряемых MATLAB-алгоритмов;
- улучшена функциональность для создания S-функций в виде M-файлов.

Данные отличия Simulink 6 от Simulink 5.1 практически не влияют на практику подготовки моделей и приемы математического моделирования, описанные в этом и последующих уроках. Поэтому их материал в равной мере относится как

к Simulink 5.1, так и к Simulink 6. В конце этого урока мы обсудим главные новинки Simulink 6 более подробно.

2.1.6. Новые возможности Simulink 7

В новейшей версии MATLAB 2007b, поступившей на рынок осенью 2007 г., применена версия пакета расширения Simulink 7, открывшая новую ветвь развития этого пакета. В Simulink 7 введен ряд усовершенствований:

- значительное ускорение моделирования при работе с новыми ПК на основе многоядерных микропроцессоров, например Intel Core 2 Duo и Quad;
- ускоритель Simulink 7 имеет новый режим Rapid Accelerator, обеспечивающий ускоренную генерацию объектного кода;
- профилировщик Simulink 7 обеспечивает идентификацию критических параметров выполнения моделирования;
- при работе с ускорителем модель может ссылаться на цели моделирования для определения используемого уровеня оптимизации компиляторов;
- расширенный переменный шаг дискретного решающего устройства исключает ненужные шаги в процессе моделирования;
- введены некоторые опции в параметры блоков, обеспечивающие более эффективное и быстрое моделирование;
- обеспечена поддержка для сложных параметров, заданных в формате с фиксированной точкой;
- улучшены интерфейсы для работы с блоками и задания их параметров;
- введены новые параметры блоков для определения минимума проекта и максимума для параметров и сигналов;
- введена новая проверка диапазона параметра в течение времени редактирования и диаграмма модификации;
- реализована новая диагностика для проверки диапазона сигнала в течение моделирования и обнаружения непрерывного типового времени на сигналах, не использующих операции с плавающей точкой;
- улучшены система параметров блоков, диагностика блоков и контроль связей библиотечных модулей в моделях.

Эти (и некоторые другие) отличия Simulink 7 от Simulink 6 носят внутренний характер и никоим образом не влияют на применение данной версии этого пакета по сравнению с применением предшествующих версий. Они просто означают, что моделирование в Simulink 7 происходит более быстро, более точно и более надежно, чем при применении Simulink 6.*. Весь последующий материал данной книги в одинаковой мере применим к версиям Simulink 6 и 7 и, с некоторыми оговорками, касающимися немного различных наборов блоков, к Simulink 5. Если читатель работает с Simulink 5, ему надо внимательно относиться к таким оговоркам.

2.2. Запуск Simulink и основы работы с пакетом

2.2.1. 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. При нажатии этой кнопки открывается окно интегрированного браузера библиотек, показанное на рис. 2.1.

Нетрудно заметить, что пользовательский интерфейс окна браузера выполнен в общем стиле, характерном для **Проводника** Windows 95/98/2000/XP. Это позволяет отказаться от детального описания его особенностей. Отметим лишь главные возможности работы с браузером.

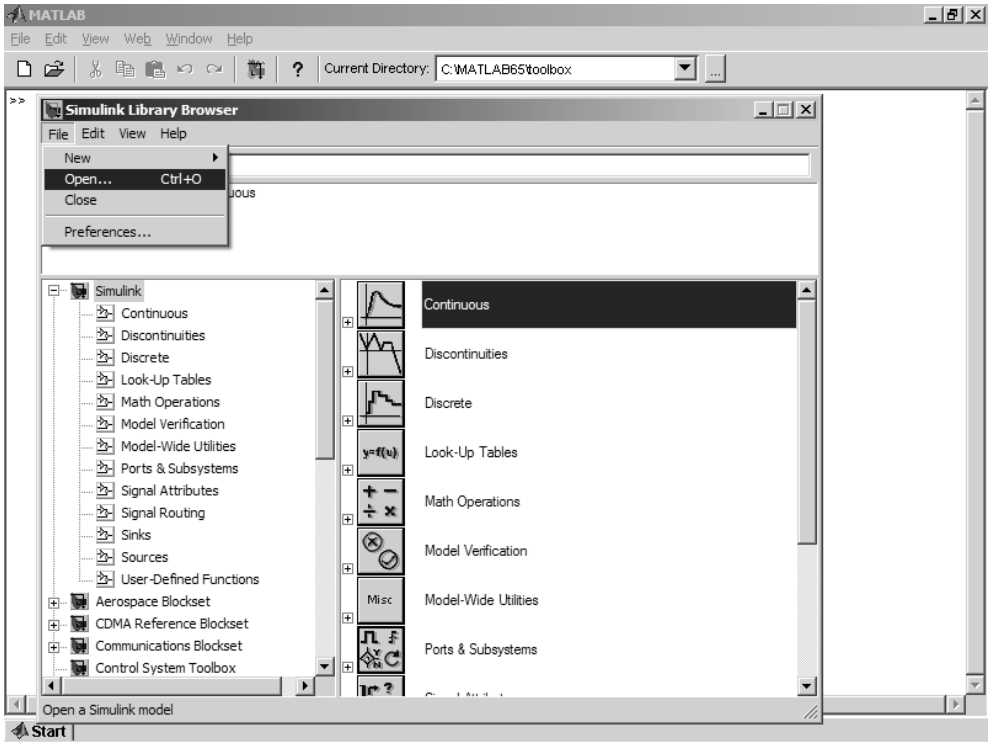


Рис. 2.1. Окно браузера библиотек Simulink 5.1

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. У версии Simulink 5.0 число разделов библиотеки возросло с 8 до 10 по сравнению с Simulink 4. Новыми являются подразделы верификации моделей Model Verification и построения субсистем Subsystems. Но уже в версии 5.1 число разделов библиотеки выросло до 13.

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.

Интеграция пакета Simulink с системой MATLAB имеет глубокий смысл. Прежде всего надо отметить, что любые манипуляции с диаграммой модели (например, ввод блоков, создание и удаление линий между ними и т. д.) выполняются соответствующими командами системы MATLAB. Их описание можно найти в разделе справки **Simulink ⇒ Function** – **By Category**, а также в [33].

Эти команды (функции) делятся на четыре категории:

- Model Construction – конструирование блоков (задание, уничтожение и сравнение блоков, операции с соединительными линиями и т. д.);
- Simulation – управление процессом моделирования;
- Linearizarion and Trimming – построение и применение линеаризированной модели;
- Data Type – работа с типами данных.

Однако знать эти команды подавляющему большинству пользователей нет необходимости, поскольку они используются автоматически в программе построения диаграммы (модели) и управления процессом моделирования. Она обычно называется *S-функцией* и генерируется автоматически. Это является важным достоинством визуально-ориентированного программирования в среде MATLAB + Simulink.

Решение большинства задач моделирования базируется на автоматическом составлении сложных конечно-разностных систем для линейных, нелинейных и дифференциальных уравнений, называемых уравнениями состояния модели. Все это также обеспечивает пакет Simulink автоматически. Наиболее эффективное решение подобных 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).

2.2.2. Запуск моделей Simulink из среды MATLAB

Обычно Simulink запускается соответствующей кнопкой из панели инструментов окна MATLAB (см. описание 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 для версии MATLAB R13 SP1 с Simulink 5.1 будет дан в следующей форме (дано только начало списка):

```
>> help simulink
  Simulink
  Version 5.1 (R13SP1) 22-Aug-2003
  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
.....................................................
```

В более новой версии MATLAB R2007a исполнение этой команды также приводит к выдаче списка команд. Ниже приведено лишь начало списка:

```
>> help simulink
Simulink
  Version 6.6 (R2007a) 01-Feb-2007
   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
    . . . . . .
```

Имена команд представлены в виде гиперссылок, и их активизация выводит в командном режиме справку по соответствующей команде.

Приведенные выше команды используются для обеспечения связи (интерфейса) между двумя, в сущности разными, программами – MATLAB и Simulink. Практически они используются автоматически в ходе визуально-ориентированного проектирования моделей. Однако опытные пользователи могут воспользоваться данными командами при решении специфических задач применения Simulink.

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.

2.2.3. Особенности интерфейса Simulink

Интерфейс Simulink полностью соответствует стилю интерфейса типичных приложений Windows 95/98/NT/2000/XP/Vista, 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/XP. Меню Simulink содержит следующие позиции:

- **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.

2.3. Работа с демонстрационными примерами

2.3.1. Поиск и загрузка модели аттрактора Лоренца

Даже незнакомый с Simulink пользователь может быстро оценить возможности этого пакета, воспользовавшись интересными и поучительными примерами, входящими в поставку Simulink. Они находятся в папке **MATLAB/TOOLBOX/SIMULINK/ SIMDEMOS**. Попутно отметим, что в папке **MATLAB/SIMULINK** располагаются служебные файлы.

Для загрузки примеров, наряду со вкладкой **Demos** справки, используется (как обычно) команда **Open** в меню **File** браузера библиотеки Simulink – рис. 2.1. В окне с названием **Open** (рис. 2.2) надо войти в указанную папку и выбрать подходящий пример.

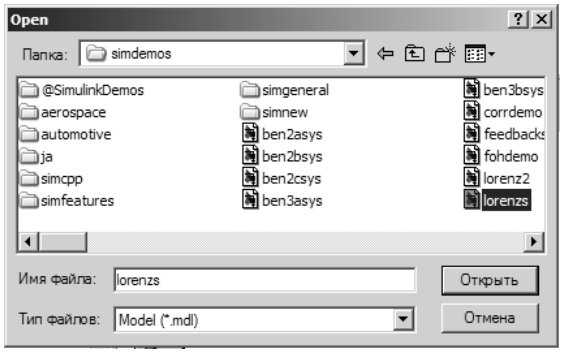


Рис. 2.2. Окно с перечнем файлов демонстрационных примеров пакета Simulink

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.

2.3.2. Установка параметров компонентов модели

Для пользователя, имеющего хотя бы начальное представление о моделировании, вполне ясно, что любой компонент модели имеет какие-то параметры или хотя бы

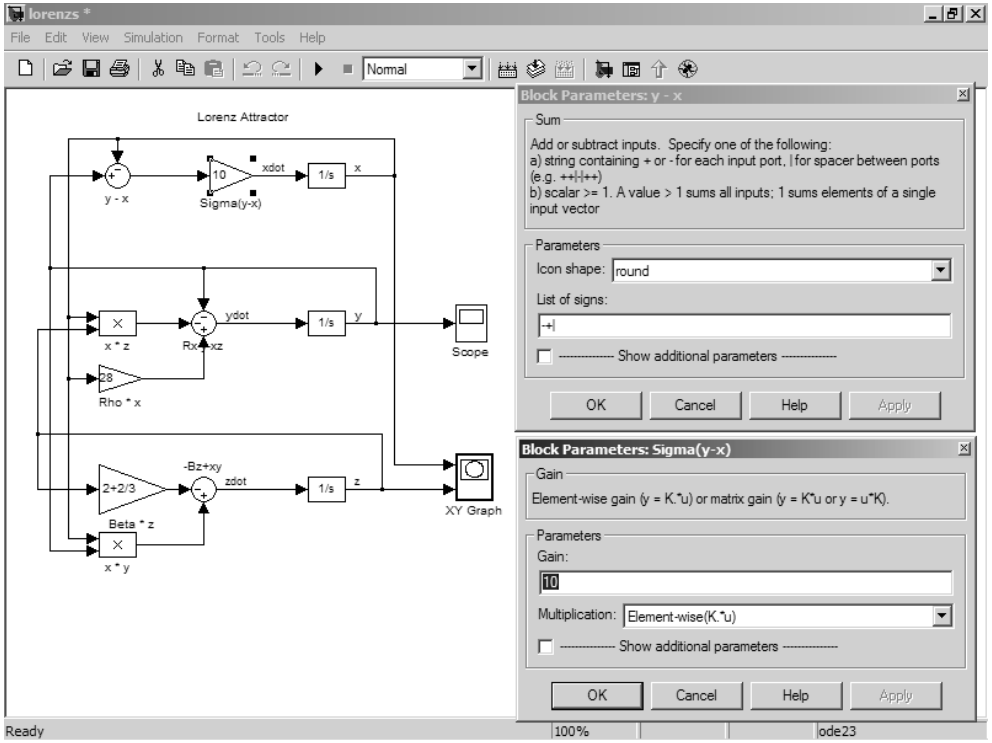


Рис. 2.3. Модель аттрактора Лоренца в окне редактирования пакета Simulink

один параметр (впрочем, есть компоненты и без параметров). 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.

2.3.3. 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.

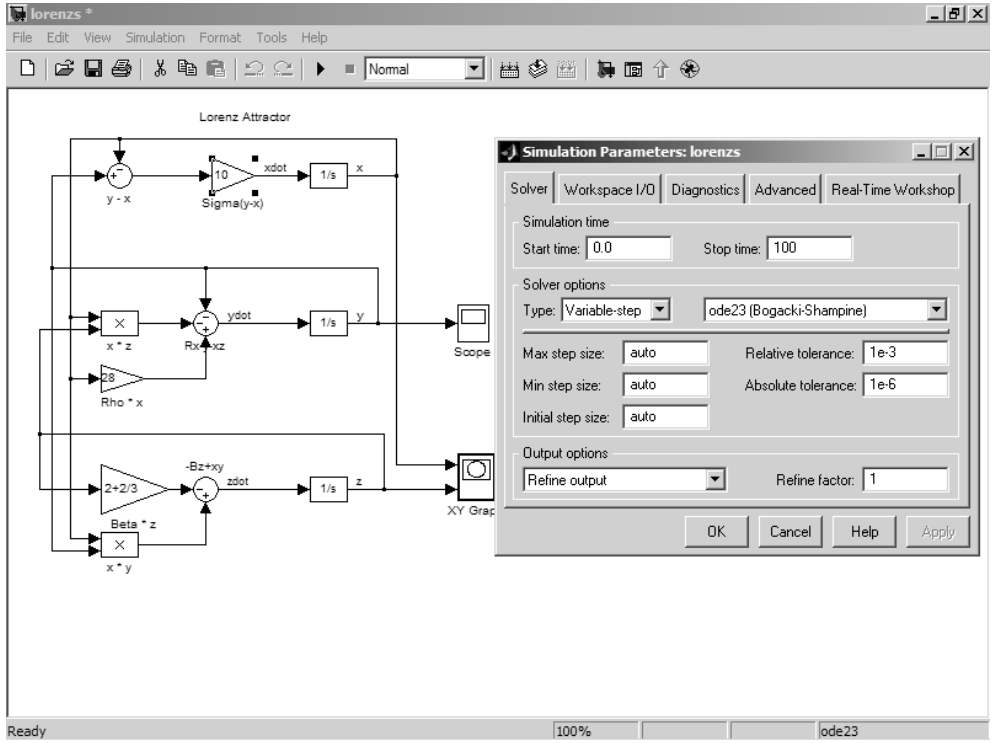


Рис. 2.4. Установка параметров модели аттрактора Лоренца

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. В противном случае шаг времени придется сильно уменьшать для описания наиболее быстрых участков изменения результатов моделирования, что ведет за собой значительное возрастание времени моделирования.

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

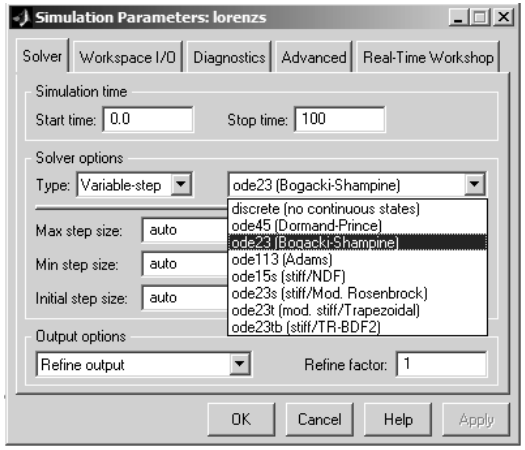


Рис. 2.5. Установка метода моделирования

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. Подробное описание указанных методов решения дифференциальных уравнений можно найти в уроке 8.

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 слишком 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.

2.3.4. 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.

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.

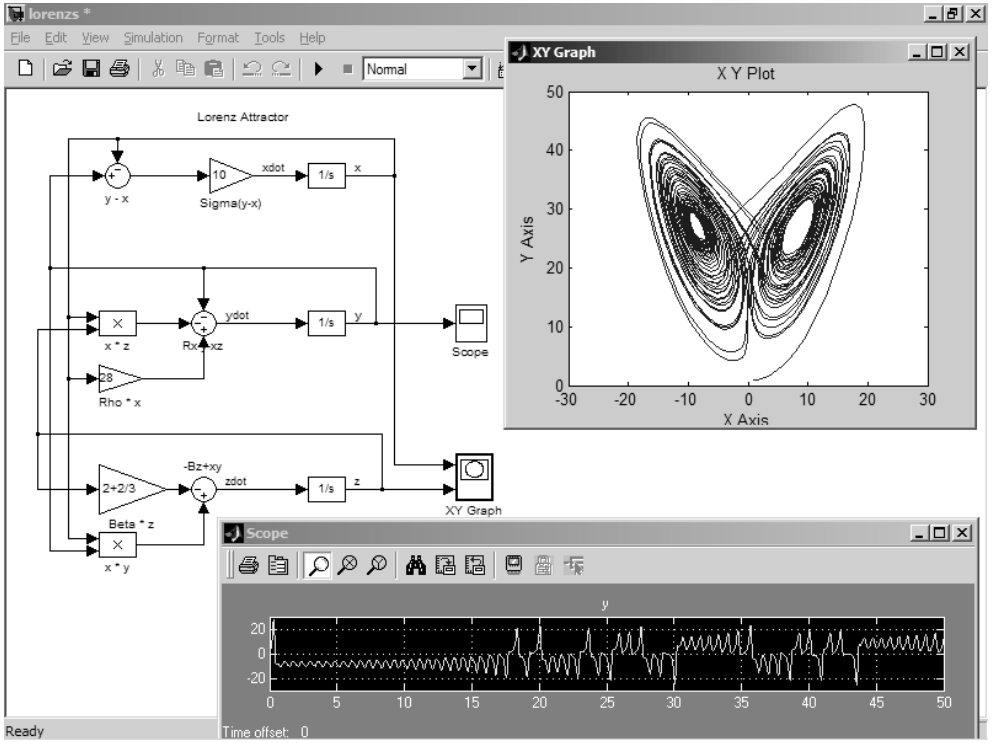


Рис. 2.6. Результаты моделирования аттрактора Лоренца

2.3.5. Решение дифференциальных уравнений 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 подобных 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

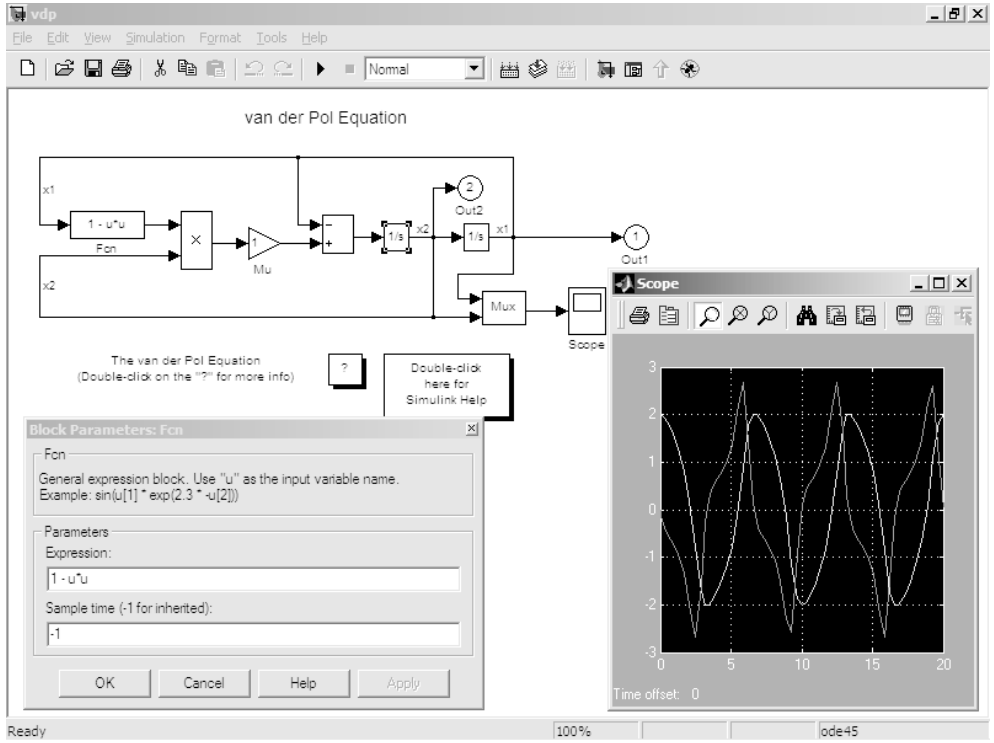


Рис. 2.7. Модель колебательной системы второго порядка

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. Они представляют собой периодические колебания, форма которых заметно отличается от синусоидальной, что является следствием нелинейности моделируемой системы.

2.3.6. Изменение характера нелинейности модели

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.

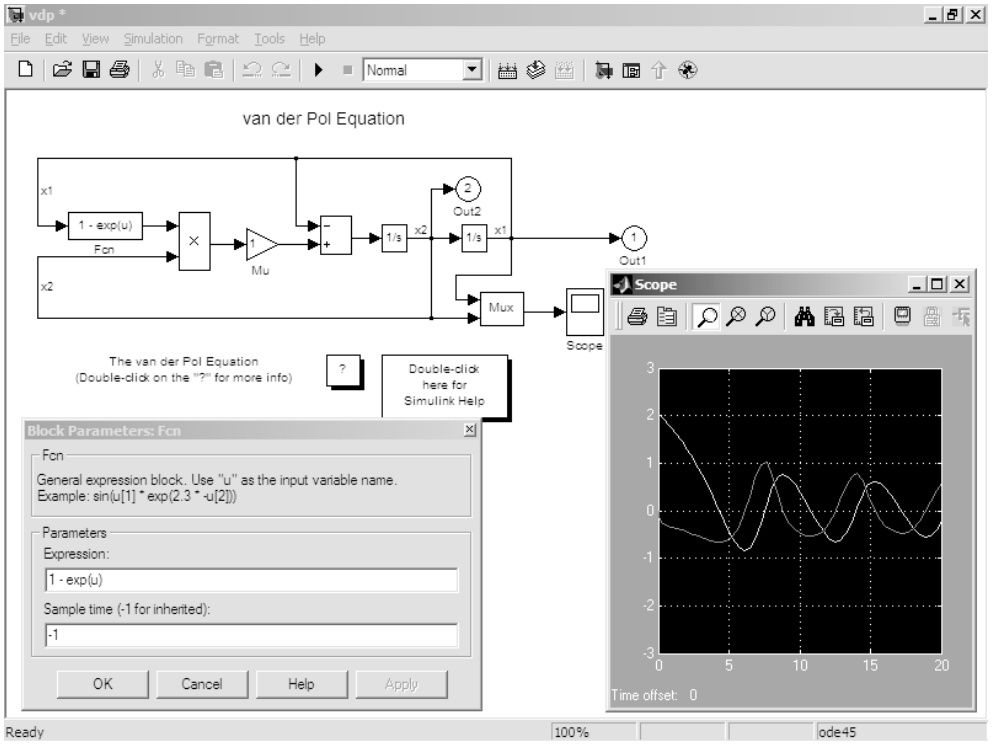


Рис. 2.8. Изменение функции нелинейности в модели системы второго порядка

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 рис. 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.

2.3.7. Как добавить в модель графопостроитель

Иногда поведение системы второго порядка удобно представить фазовым портретом колебаний – как в нашем первом примере (рис. 2.6). Фазовый портрет двух

временных зависимостей (например, напряжения и тока в электрической цепи) строится в виде параметрически заданного графика. В случае его построения осциллографом или графопостроителем на вход Y подается сигнал одной временной зависимости, а на вход X – другой.

В нашем 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.

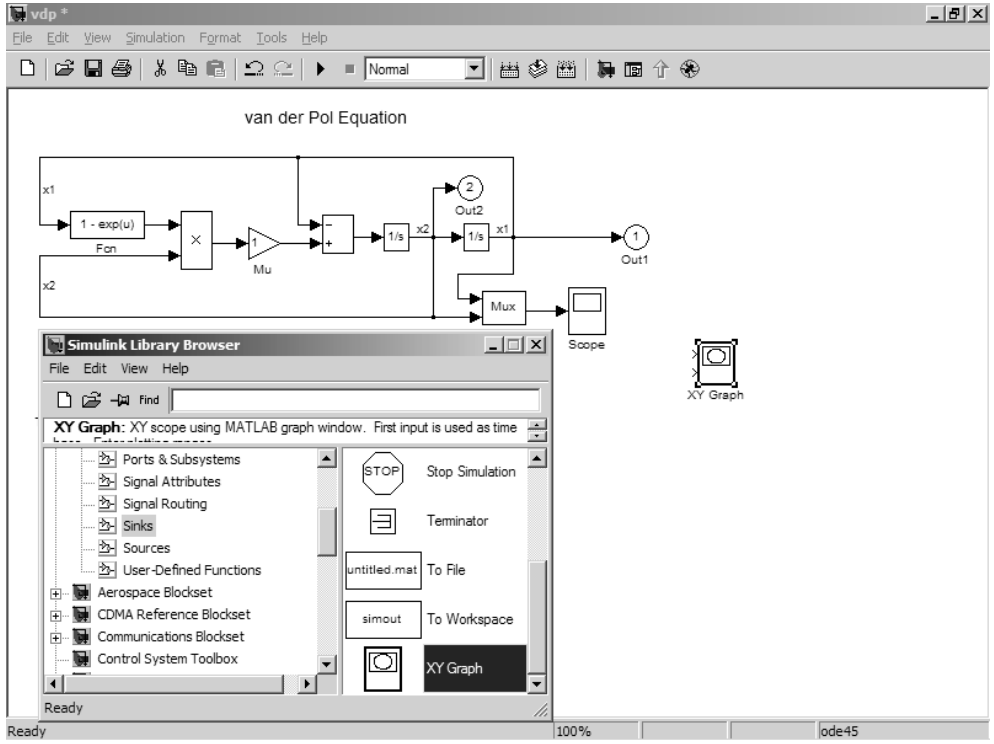


Рис. 2.9. Перенос графопостроителя в модель

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

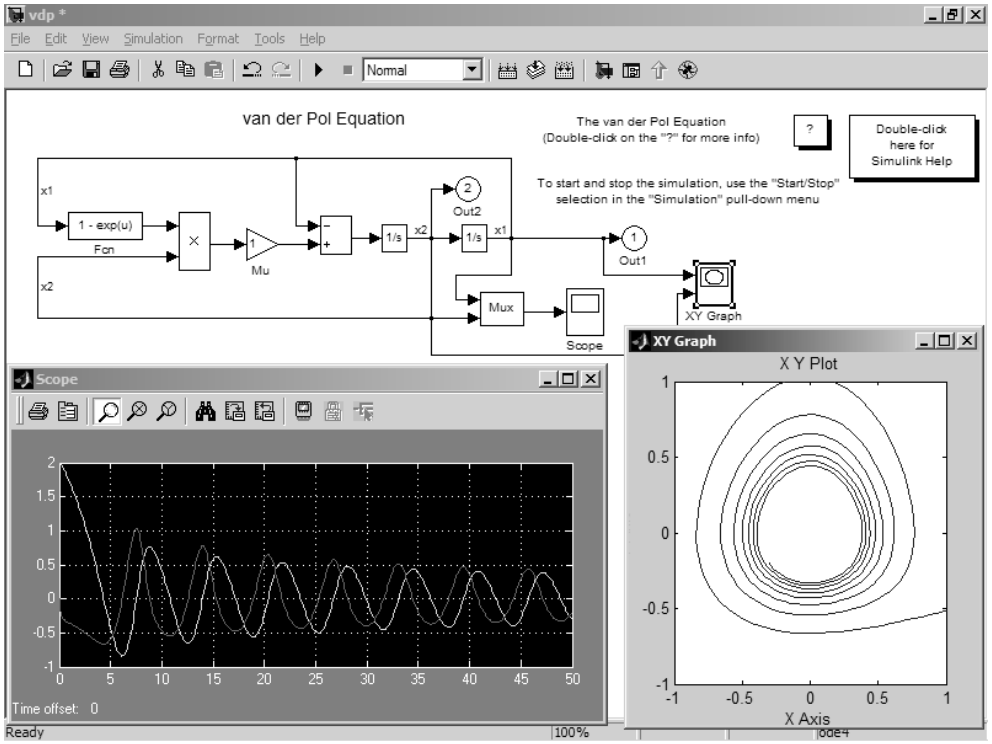


Рис. 2.10. Модель с подключенным графопостроителем

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. Увеличим до 50 время моделирования и перейдем к моделированию с фиксированным шагом 0,1, что сделает кривые переходных процессов более плавными. Для моделирования используем решатель Рунге-Кутта.

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 оперативно 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.

2.4. Работа с редактором дифференциальных уравнений

2.4.1. 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**.

Загрузка файла приводит к появлению окна с набором демонстрационных примеров, показанного на рис. 2.11 в левом верхнем углу. 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 отображает 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.

2.4.2. 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.

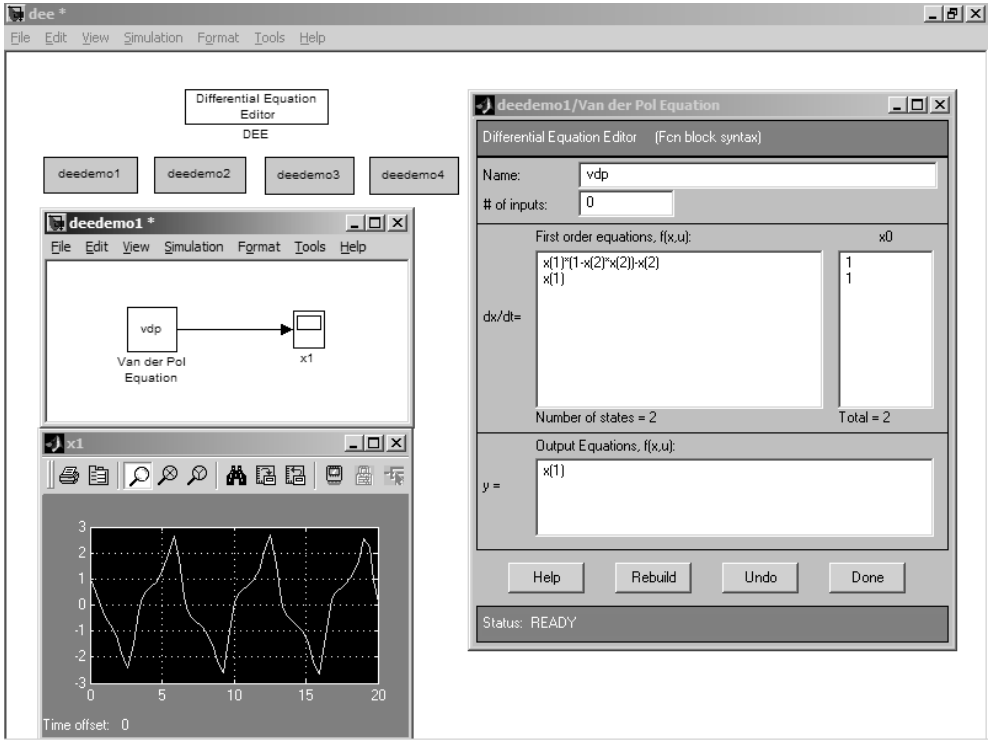


Рис. 2.11. Решение системы дифференциальных уравнений Ван-дер-Поля

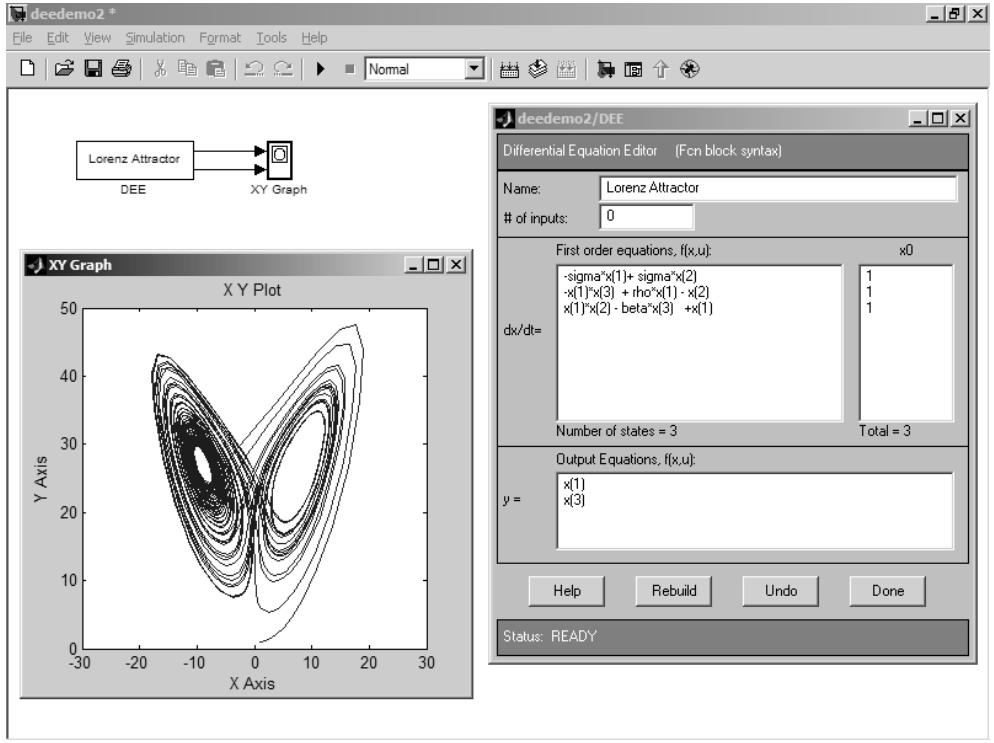


Рис. 2.12. Решение системы дифференциальных уравнений аттрактора Лоренца

2.5. Дополнительные примеры моделирования

2.5.1. Моделирование кубика с пружинкой

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 его 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.

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.

2.5.2. 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 аналогична описанной выше.

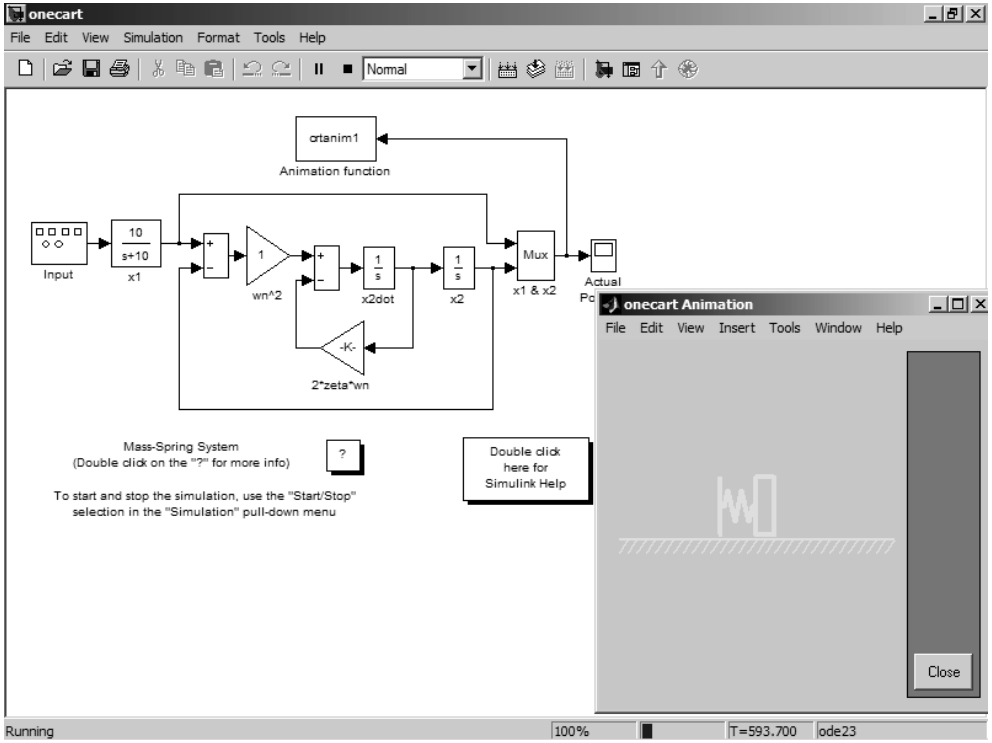


Рис. 2.13. Модель кубика с пружинкой

2.5.3. Моделирование системы терморегулирования дома

Рисунок 2.15 показывает модель системы терморегулирования для отдельного дома. Это один из демонстрационных примеров, входящих в справку по пакету Simulink.

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.

2.5.4. Использование субмоделей

В этом примере мы сталкиваемся с новой принципиально важной и эффективной возможностью пакета Simulink – использованием субмоделей (подсистем). Субмодель строится практически так же, как и модель системы. Она имеет обозначенные цифрами порты входа и выхода, через которые соединяется с основной моделью.

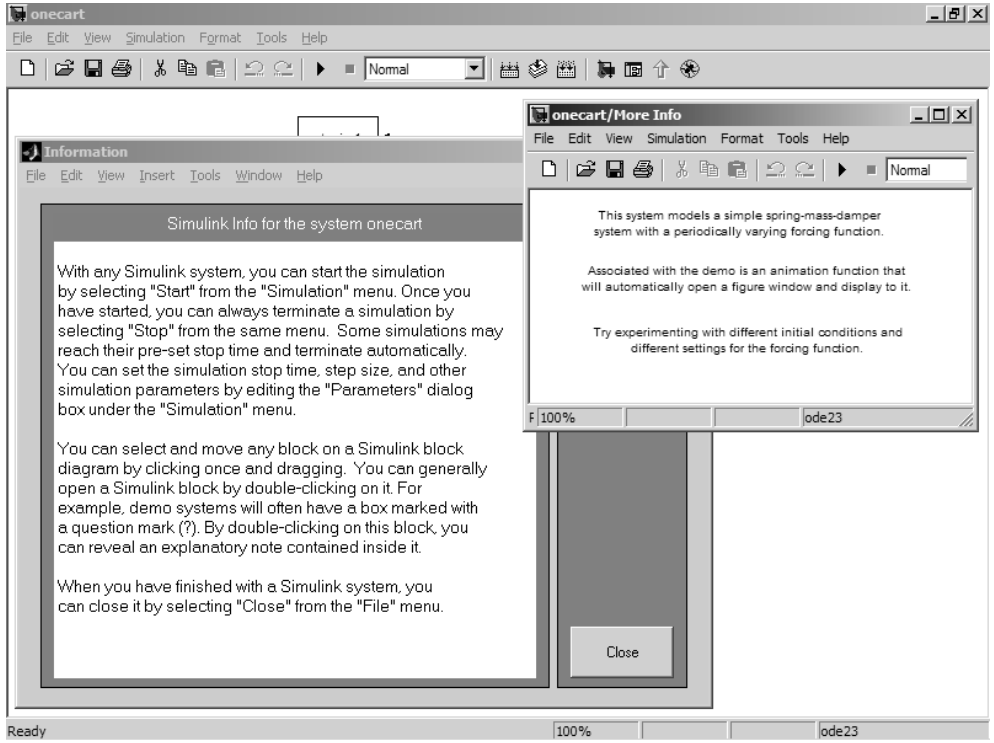


Рис. 2.14. Активизация гиперссылок

На рис. 2.15 показан браузер субмоделей, который был введен еще в Simulink 4. 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.

2.5.5. Моделирование работы 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. Это типичная модель импульсной гидравлической системы релаксационного типа с внешним запуском – ручкой слива воды из сливного бачка.

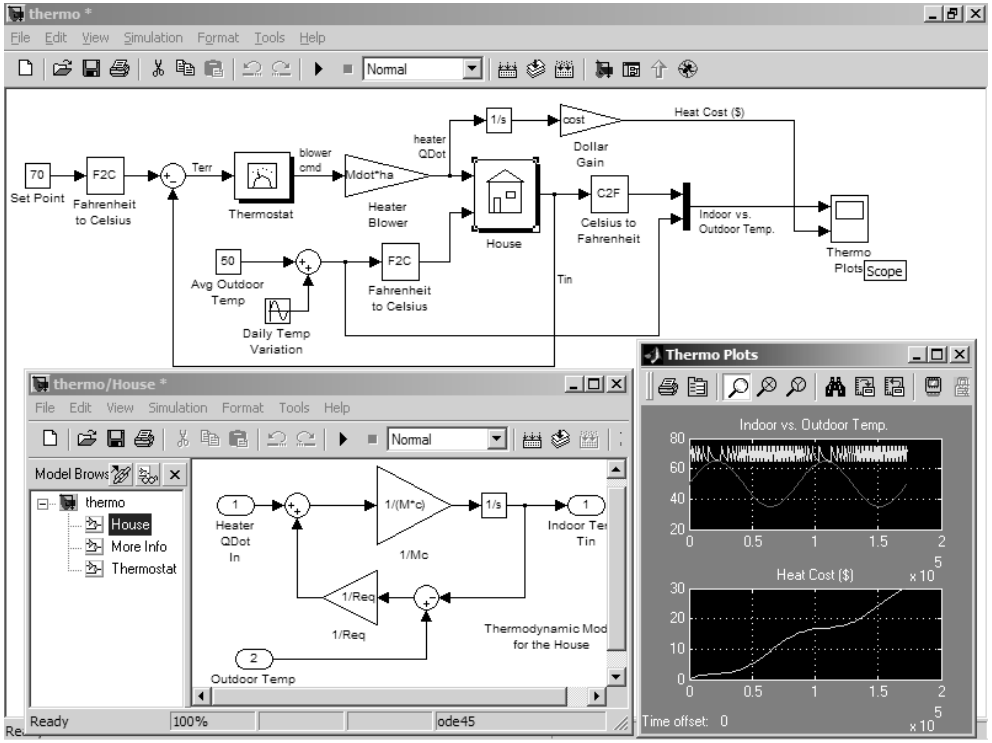


Рис. 2.15. Система терморегулирования дома

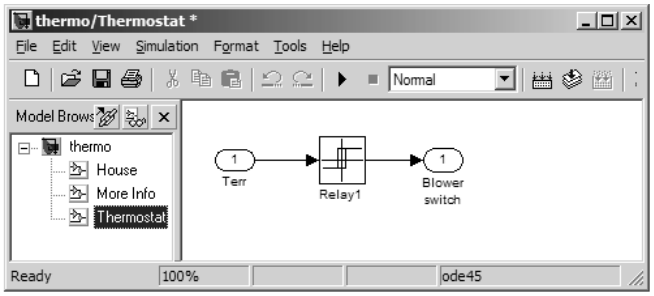


Рис. 2.16. Субмодель термостата

В данном 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.

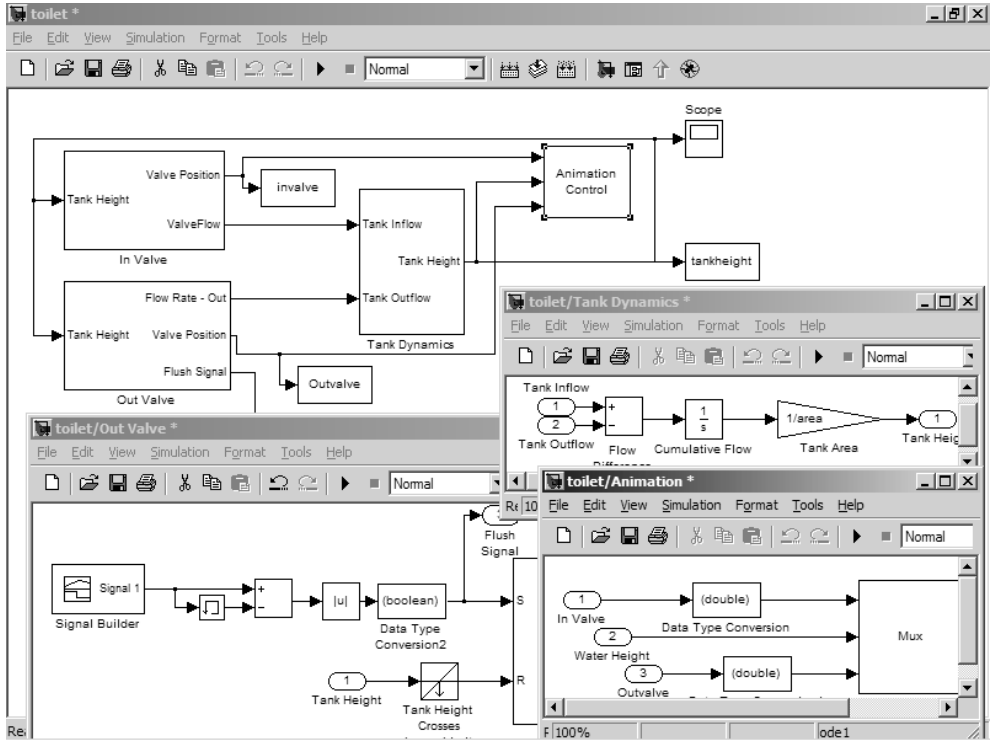


Рис. 2.17. Модель сливной системы унитаза

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.

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.

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

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.

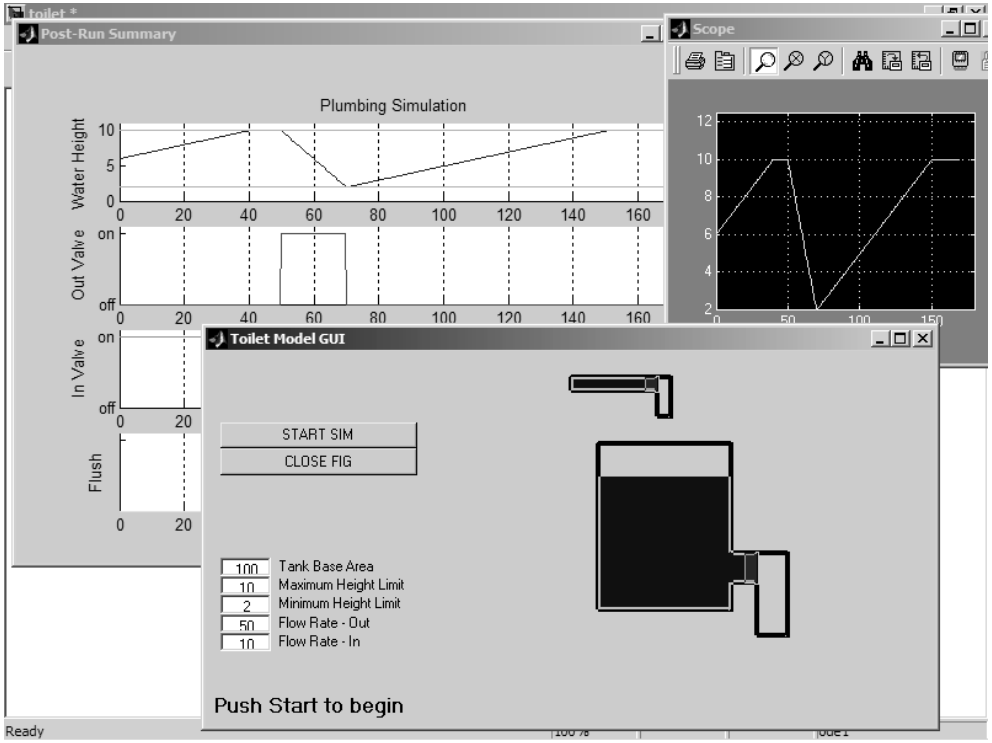


Рис. 2.18. Средства визуализации модели сливной системы унитаза

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 происходят наполнение бака водой и ее слив в нужный момент.

2.6. Дополнительные возможности

2.6.1. Применение логических операций

Многие системы и устройства содержат блоки логических операций, которые подробно описывались в уроке 1. Рисунок 2.19 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.

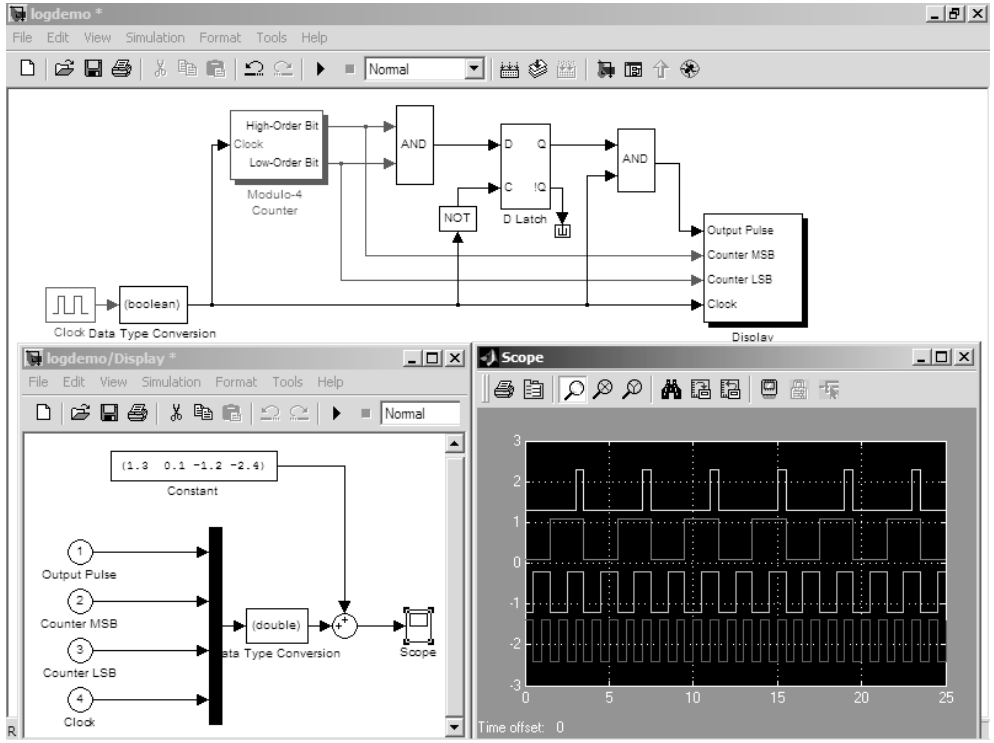


Рис. 2.19. Модель генератора трех последовательностей прямоугольных импульсов

2.6.2. 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. 2.20), 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.

Пример, приведенный на рис. 2.20, весьма поучителен. Он показывает, насколько выбор данных может повлиять на результат вычислений. Помимо обрезания верхушек синусоиды, из-за ограничения чисел типа unit8 можно наблюдать обрезание нижних полуволн синусоиды при использовании данных Unsignet (числа без знака). Это говорит о том, что в выборе специальных данных и при их преобразованиях нужно соблюдать известную осторожность.

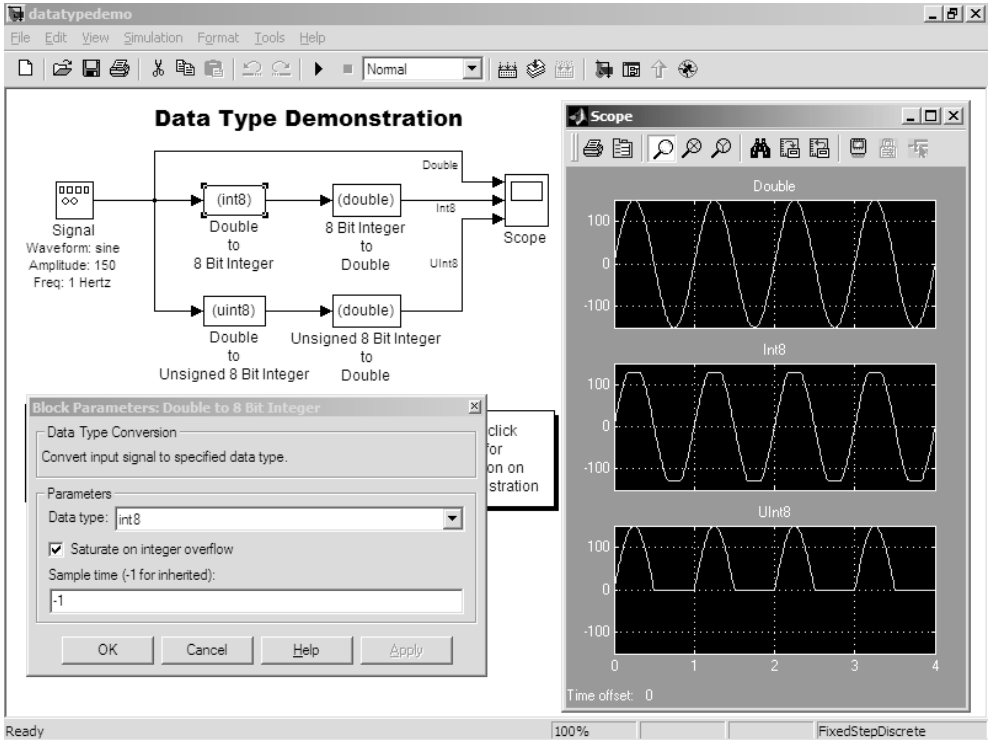


Рис. 2.20. Модель с указанием типов примененных в ней данных

Рис. 2.21. Окна браузера библиотек Simulink 6.6 (MATLAB R2007a)

2.7. Особенности реализации Simulink 6

2.7.1. Новые разделы библиотеки Simulink 6

Версия Simulink 6 отличается от Simulink 5 некоторой доработкой библиотеки. Окно браузера библиотек Simulink 6 представлено на рис. 2.21 для версии Simulink 6.6 системы MATLAB R2007a.

Общее представление о разделах библиотеки Simulink 6 дает главное окно библиотек, представленное на рис. 2.22. В нем показаны пиктограммы всех разделов библиотеки. В нем новые разделы библиотеки представлены снизу отдельной группой. Это облегчает работу с библиотекой тем пользователям, которые сменили версию Simulink 5 на Simulink 6.

Первое, что бросается в глаза, – это наличие нового большого раздела библиотеки Commonly User Blocks, в который вынесены наиболее часто применяемые пользователями блоки. Самостоятельного значения этот раздел не имеет, по-

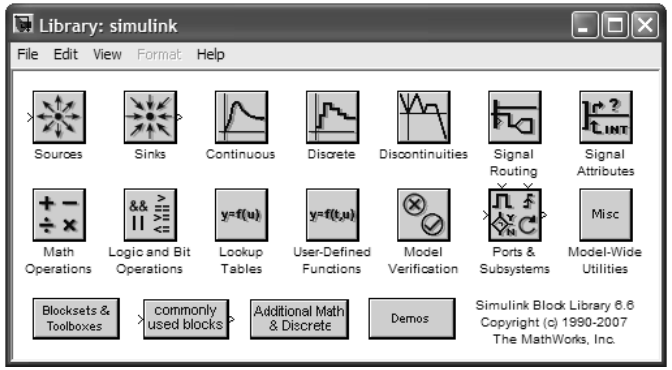


Рис. 2.22. Главное окно библиотек Simulink 6.6

скольку его блоки есть в более специальных разделах библиотеки. Тем не менее этот блок удобен при моделировании наиболее типичных задач.

В Simulink 6 введены еще два новых раздела библиотеки блоков:

- Logic and Bit Operation – логические и побитовые операции;
- Additional Math & Discrete – расширенные математические и дискретные блоки.

2.7.2. Подборка блоков из ящиков Blockset и Toolbox

В ряде пакетов расширения из ящиков Blockset и Toolbox имеются свои блоки для построения ряда специальных моделей, например механических или электрических устройств и т. д. Они разбросаны по многочисленным пакетам расширения, что затрудняло построение комбинированных моделей устройств. В Simulink 6 эта трудность была преодолена – в библиотеке появился раздел Blocksets & Toolboxed, в котором собраны такие блоки. Окно этого раздела показано на рис. 2.23.

Пакет Simulink 6.* интенсивно развивается, и в различных его вариантах возможны непринципиальные (для его изучения и применения) отличия в виде окон интерфейса и их содержимого. Так, в новейшей версии Simulink 6.6 от раздела

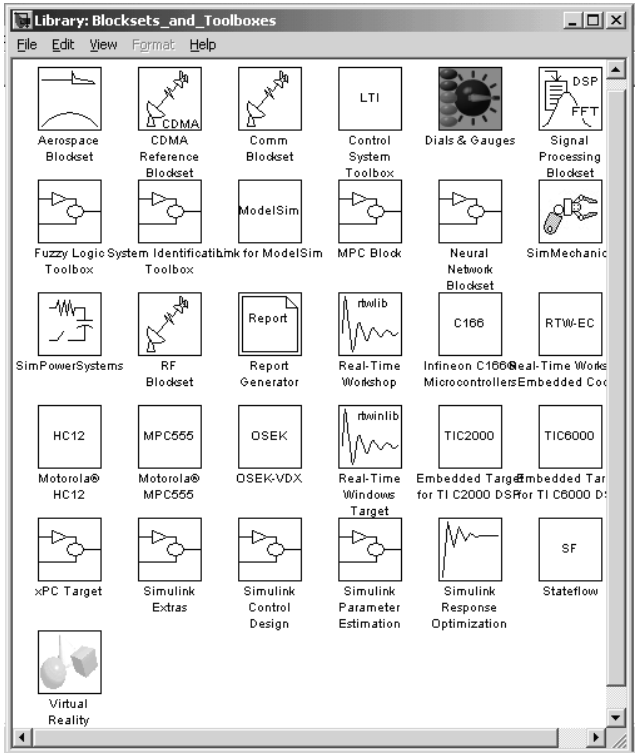


Рис. 2.23. Окно раздела библиотек Simulink 6 Blocksets & Toolboxed

Blocksets & Toolboxed уже отказались. В связи с этим материал данной книги не претендует на абсолютно точное описание деталей интерфейса всех версий и подверсий Simulink, просто невозможное в рамках объема данного самоучителя.

2.7.3. Новое окно установки параметров моделирования

Окно установки параметров моделирования Simulink 5 выполнено в несколько старомодной манере – рис. 2.4 справа. В Simulink 6 это окно осовременено и показано на рис. 2.24 (снизу) для случая моделирования с переменным шагом.

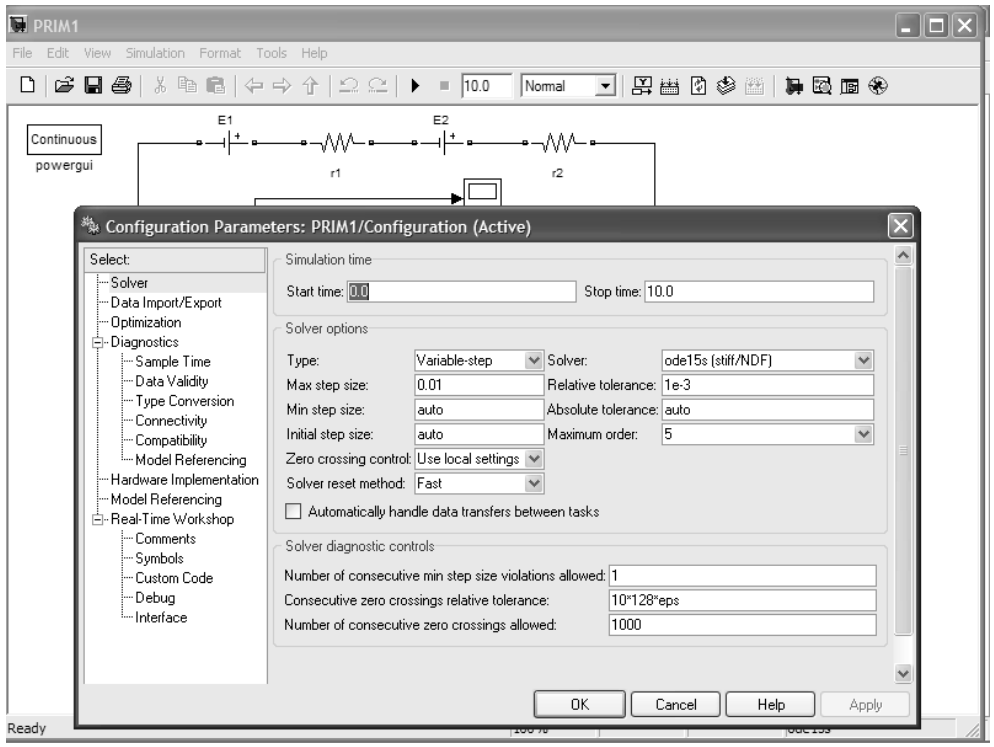


Рис. 2.24. Окна установки параметров моделирования Simulink 6.6 с переменным шагом

Мы не будем рассматривать все 18 вкладок этого окна. Отметим, что по сути (а не по внешнему оформлению) содержимое главной вкладки Solver (установка параметров решателей дифференциальных уравнений) ничем не отличается от аналогичного для подобной вкладки окна установки параметров Simulink 5. Заметим лишь, что вид вкладки различный для разных видов моделирования – с переменным шагом (рис. 2.24) и фиксированным шагом (рис. 2.25).

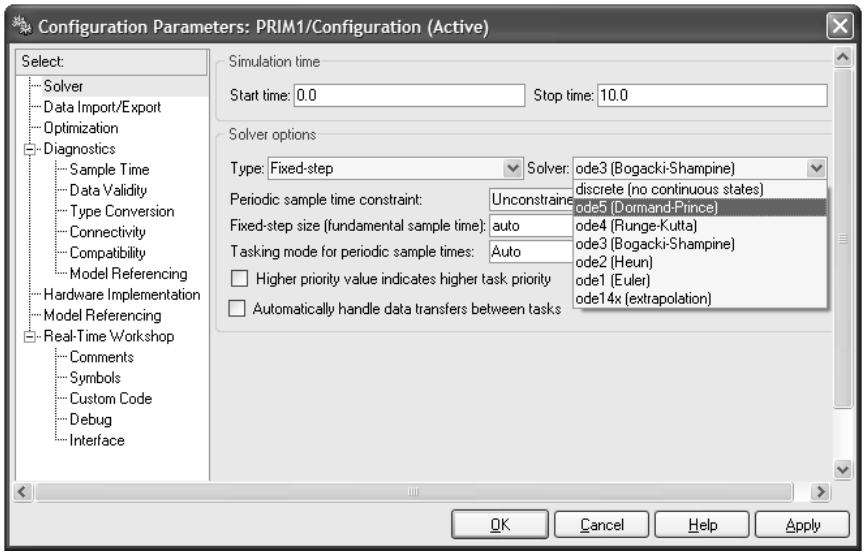


Рис. 2.25. Окна установки параметров моделирования в Simulink 6.6 с фиксированным шагом

С назначением установок на других вкладках нетрудно ознакомиться самостоятельно. Большое число вкладок позволило разнести установки по характерным для них признакам.

2.7.4. Новые кнопки на панели инструментов Simulink

Небольшой доработке в Simulink 6 подверглась панель инструментов. В ней появились новые кнопки:

- **Ready** – считывание данных;
- **Refresh Model blocks** – обновление (перерисовка) модели;
- **Launch Model Explorer** – вывод окна навигатора модели.

Применение новых кнопок облегчает работу с Simulink 6.

2.7.5. Новый навигатор моделей Model Explorer

Из новых средств Simulink 6 стоит выделить навигатор модели Model Explorer. Он запускается кнопкой Launch Model Explorer в панели инструментов. Окно навигатора модели показано на рис. 2.26.

Навигатор модели дает полную информацию обо всех блоках модели. В левой части окна навигатора представлено дерево модели, в середине – список всех блоков, и справа – окно параметров, или описание выделенного блока. При отладке сложных моделей возможности навигатора трудно переоценить.

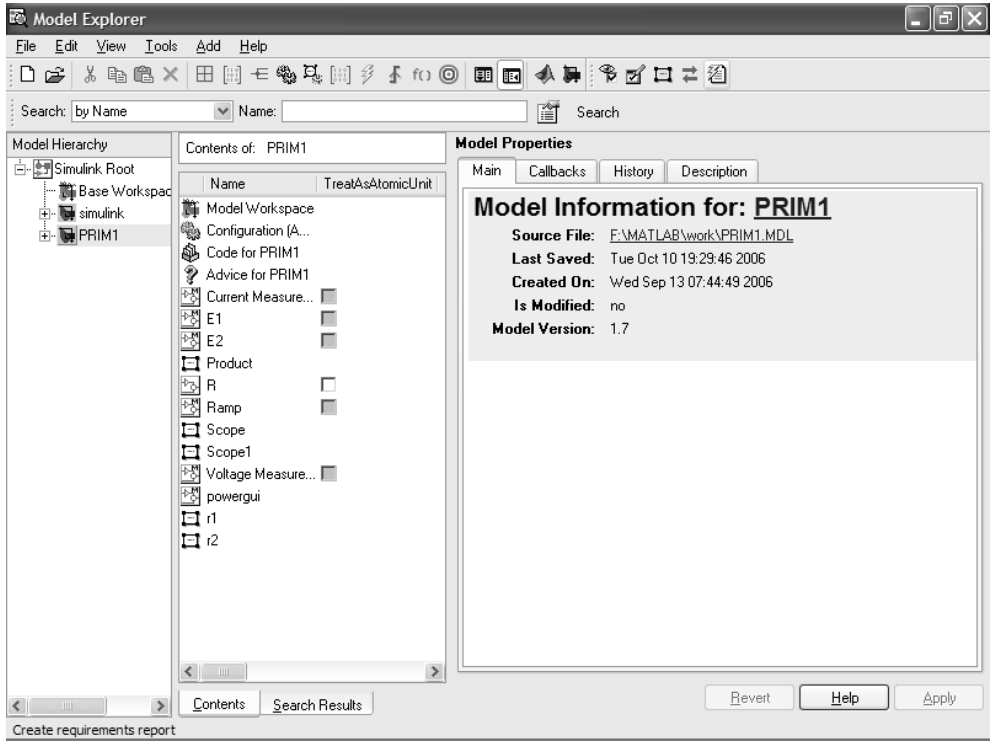


Рис. 2.26. Окно навигатора модели Simulink 6

2.7.6. Расширение меню Tools

Набор инструментов в Simulink 6.* постоянно увеличивается от версии к версии, что ведет к изменению числа позиций меню Tools. Проследить за этим в книге, подобной данному самоучителю, просто невозможно. Поэтому ограничимся представлением окна модели в последней версии Simulink 6.6 – рис. 2.27.

С многочисленными новыми возможностями команд этого меню мы познакомимся позже, по мере описания возможностей Simulink. Читателю рекомендуется ознакомиться с новыми для него командами применяемой версии Simulink самостоятельно. Большинство из них открывает окна с вполне очевидными установками.

Все указанные выше нововведения Simulink 6.* и многие иные приятные мелочи его реализации практически не влияют на процесс подготовки, отладки и запуска моделей на моделирование. Поэтому последующие уроки по этому расширению в равной мере относятся к версиям Simulink 5 и Simulink 6.*. При этом почти все примеры даны для наиболее распространенной версии Simulink 6.*.

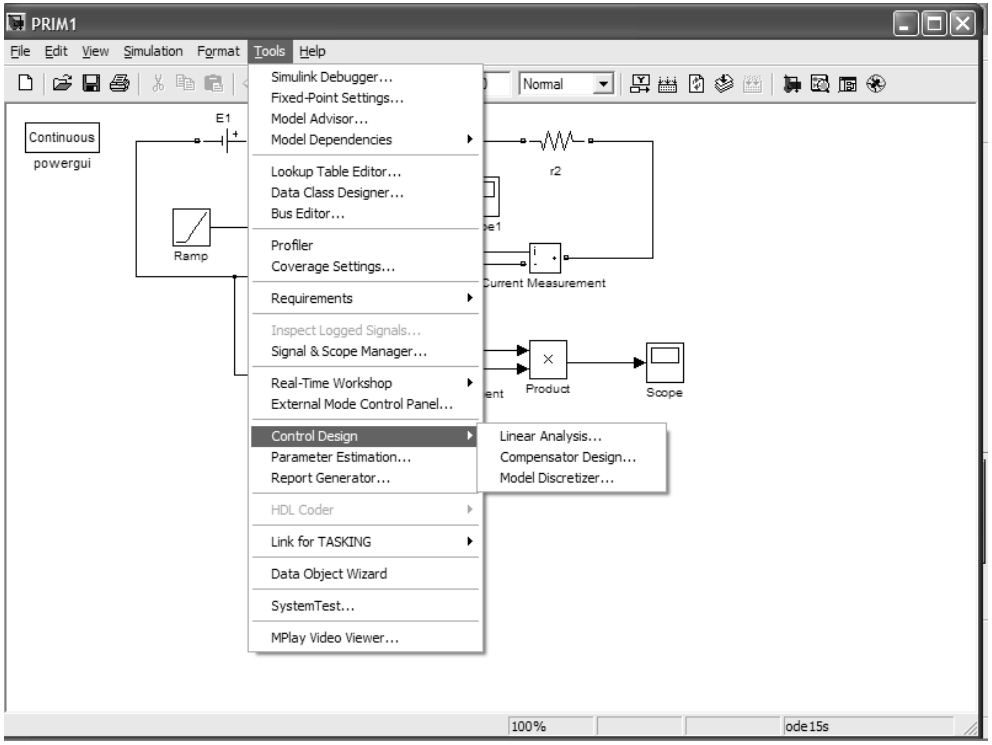


Рис. 2.27. Окно модели Simulink 6.6 с открытым меню Tools

2.7.7. Справочная система Simulink 6 и работа с ней

Справочная система (или просто справка) Simulink 6 организована по принципу построения справки по базовой системе MATLAB и интегрирована с ней. Доступ к справочным данным возможен из командной строки, из позиции **Help** меню и из инструментальных панелей (кнопка с вопросительным знаком). Кроме того, команда **Simulink Help** из позиции меню **Help** браузера библиотек (рис. 2.21) выводит окно справки MATLAB R2007a с разделом справки по Simulink 6.6. Это окно показано на рис. 2.28.

Справка (рис. 2.27) реализует доступ к нужной информации через применение гипертекстовых ссылок. Работа со справкой вполне очевидна и описана в первом самоучителе этой серии – по базовой системе MATLAB.

Спецификой справки по пакету расширения Simulink является большое число разделов с описанием блоков системы. На рис. 2.29 показан пример получения информации по блоку интегрирования. Выведено только начало описания этого блока. С помощью линейки прокрутки можно просмотреть все описание, в том числе примеры применения данного блока, описание окна установки его параметров и т. д.

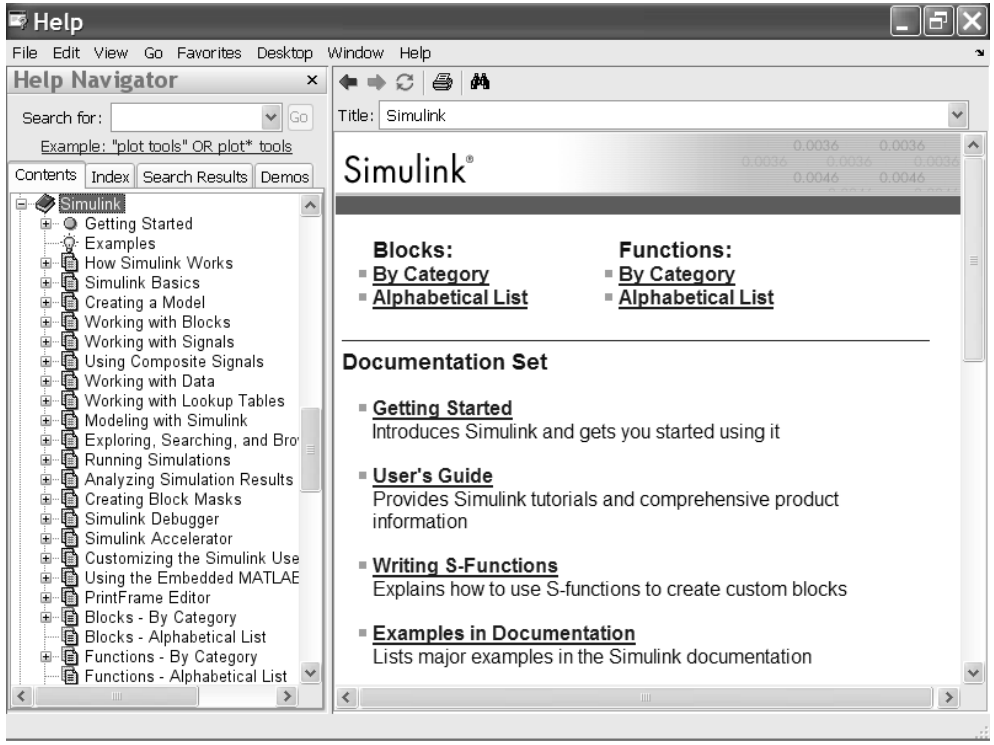


Рис. 2.28. Окно справки Simulink 6.6

2.8. Интерфейс пакета расширения Simulink 7

2.8.1. Справка по Simulink 7

Новейшая реализация пакета расширения Simulink 7 входит в состав системы MATLAB R2007b. С отличительными и новыми возможностями Simulink 7 и базовой системы MATLAB R2007b можно познакомиться по их справке. Окно справки показано на рис. 2.30.

Сравнив рис. 2.30 с рис. 2.29, нетрудно заметить, что интерфейс справки и принципы ее построения в новой реализации MATLAB не изменились. Окно рис. 2.30 соответствует разделу **Release Notes** (Замечания по реализации) и позволяет детально ознакомиться с конкретными особенностями и новыми средствами данной реализации системы, в том числе и пакета расширения Simulink 7. Они уже были отмечены в разделе 2.1.5.

При использовании справки Simulink 7 пользователь приметит характерную деталь интерфейса: при загрузке тех или иных разделов справки появляется пик-

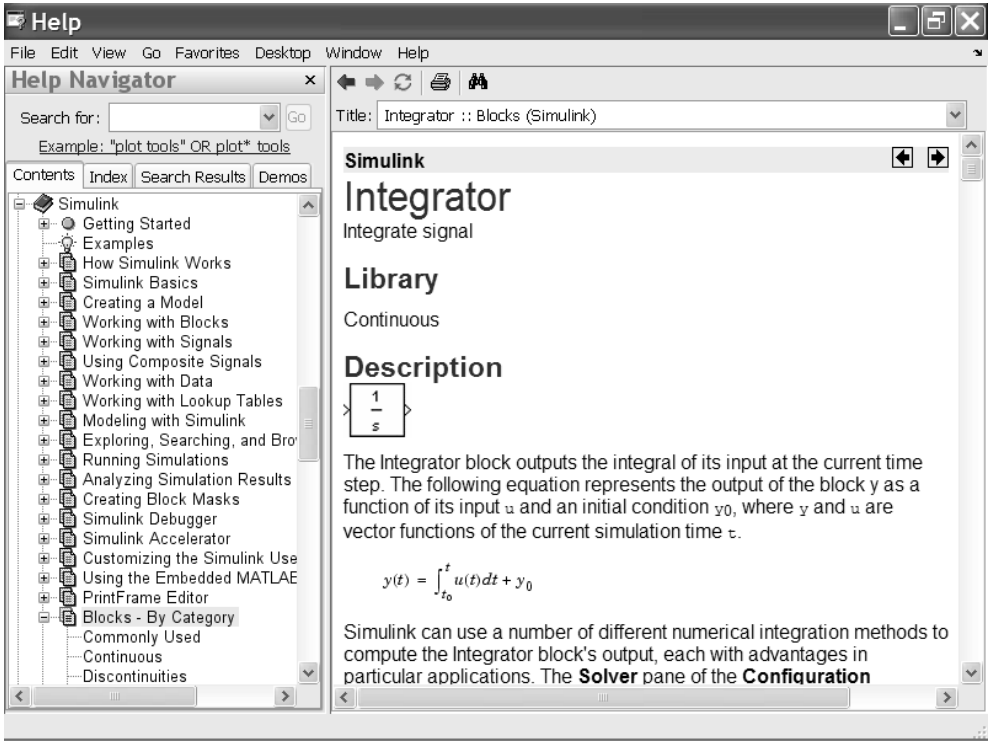


Рис. 2.29. Пример получения справки по блоку интегрирования пакета Simulink 6.6

тограмма земного шара с эмблемой (логотипом) MATLAB и надписью «Loading». При этом при использовании MATLAB R2007b на современных ПК с двух- и четырехъядерными микропроцессорами загрузка как самой системы, так и ее частей (например, разделов справки) происходит в несколько раз быстрее, чем на «старых» однопроцессорных ПК.

2.8.2. Браузер библиотек Simulink 7

Браузер библиотек новейшей версии Simulink 7 (рис. 2.31 – окно слева) практически также не отличается от такового для более ранних реализаций MATLAB R2006*/R2007a. Принадлежность его к последней версии MATLAB R2007b явно видна только при выводе окна About Simulink (рис. 2.31 – окно справа). В этом окне видна и дата реализации пакета Simulink 7 – 2 августа 2007 г.

По рис. 2.31 нетрудно заметить, что общий вид интерфейса пакета Simulink 7 не претерпел заметных изменений по сравнению с предшествующими реализа-

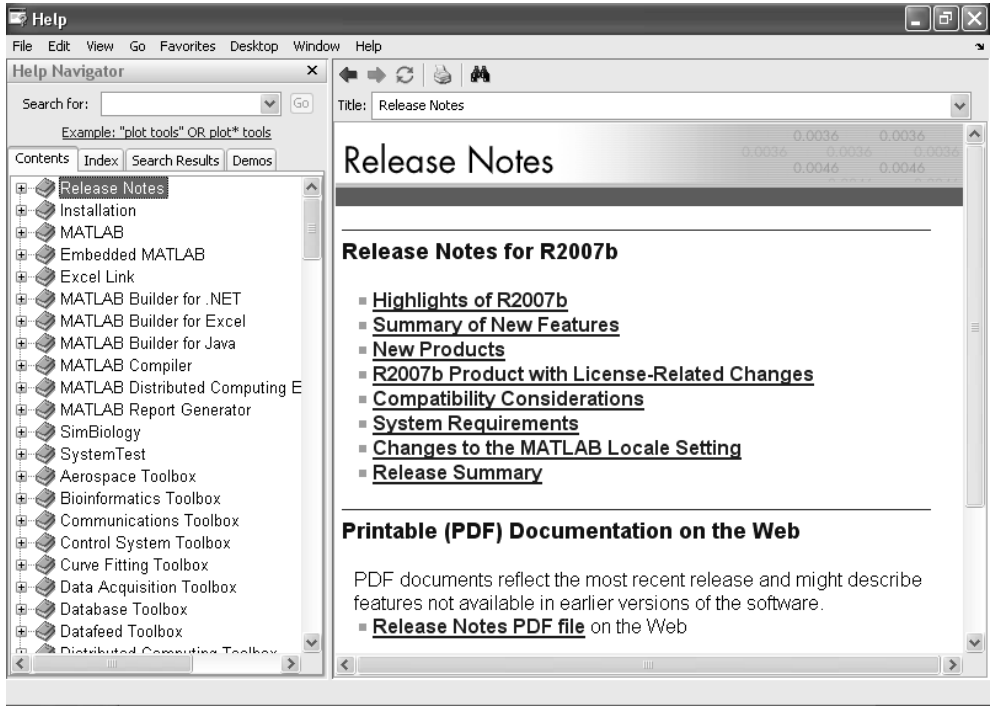


Рис. 2.30. Окно справки по системе MATLAB R2007b + Simulink 7

циями Simulink 6.*. Это обеспечивает преемственность работы с различными реализациями пакета Simulink.

2.8.3. О составе блоков библиотеки Simulink 7

Изучение новейшей реализации Simulink 7 показало, что у последней налицо и полная совместимость библиотек как на уровне основных разделов библиотеки, так и на уровне состава блоков в этих разделах. В этом нетрудно убедиться, сравнив окно с основными разделами библиотеки Simulink 7 (рис. 2.32 – слева и сверху) с таким же окном предшествующей реализации Simulink 6.6 (рис. 2.22).

Идентичность библиотек Simulink 7 и Simulink 6.* позволяет не заострять внимания пользователя на деталях моделирования, связанных с той или иной версией применяемого пакета расширения Simulink. При этом весь материал книги, представленный ниже, в одинаковой мере относится к Simulink 6* и Simulink 7.

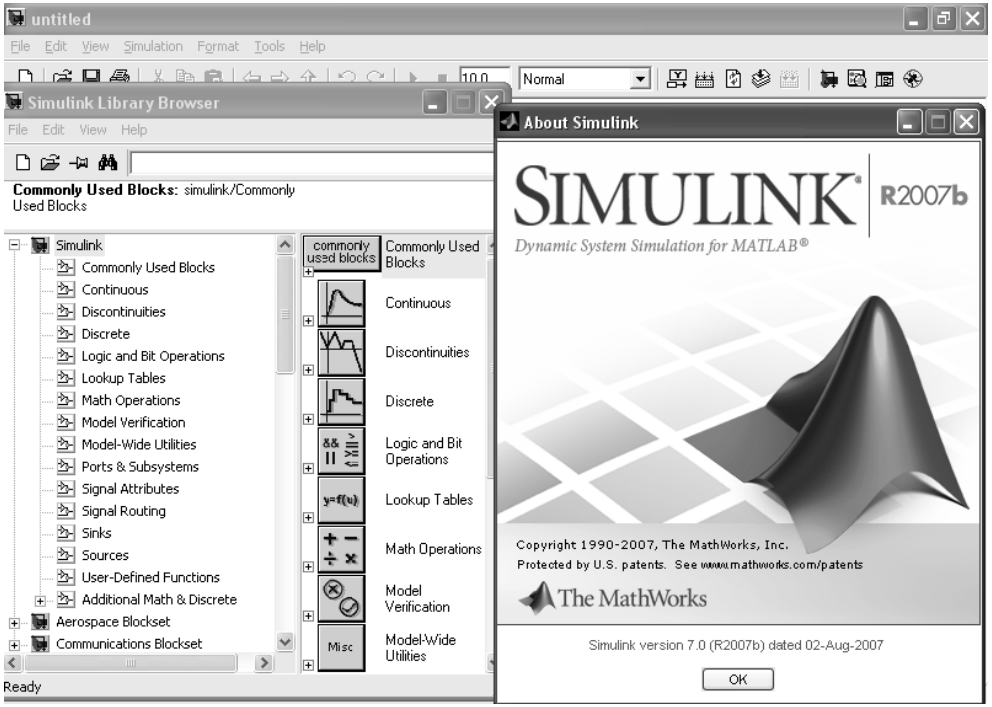


Рис. 2.31. Окно браузера библиотек Simulink 7 (слева) и окно About Simulink (справа)

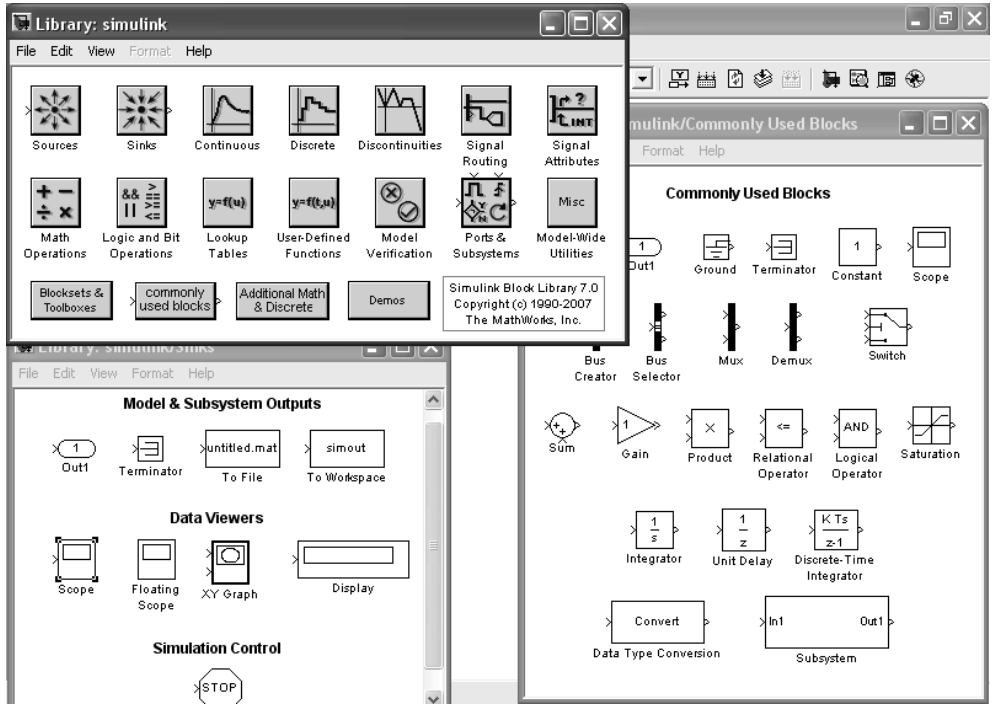


Рис. 2.32. Окна основного раздела библиотеки и окна блоков двух подразделов библиотеки Simulink 7

2.8.4. Доступ к демонстрационным примерам Simulink 7

В новых реализациях Simulink 6.* и Simulink 7 значительно усовершенствован доступ к демонстрационным примерам и резко возросло их число. Покажем осуществление доступа к таким примерам на примере последней реализации системы MATLAB R2007b + Simulink 7.

Доступ к демонстрационным примерам открывает вкладка **Demos** в окне справки по системе MATLAB R2007b + Simulink 7. Окно справки с открытой вкладкой Demos показано на рис. 2.33.

На самой вкладке **Demos** представлено дерево демонстрационных примеров с различными папками и отдельными примерами. Рисунок 2.33 иллюстрирует доступ к демонстрационному примеру на моделирование гидравлического устройства с четырьмя цилиндрами. Для открытия диаграммы (модели) этого (или другого) примера достаточно активизировать гиперссылку **Open this model**, которая видна сверху правого окна справки. Гиперссылка с именем файла позволяет ввести файл примера в окно редактора-отладчика и просмотреть в нем так называемую S-функцию модели. S-функция – это, по существу, программа, которую создает Simulink в процессе визуально-ориентированного программирования задачи на моделироваие того или иного устройства или системы.

На рис. 2.34 показано окно с диаграммой примера на моделирование гидравлического устройства с четырьмя цилиндрами, которое появляется при активизации гиперссылки для этой модели. Работа модели сопровождается появлением окон двух виртуальных осциллографов, осциллограммы которых иллюстрируют работу модели.

Для Simulink разработан ряд пакетов математического моделирования в различных сферах науки и техники. Многие из этих пакетов описаны в последующих уроках данного самоучителя. Примеры на их применение также представлены на вкладке **Demos** в папке Simulink. На рис. 2.35 показано окно справки по примерам пакета расширения SimPowerSystems, ориентированного на моделирование силовых электротехнических устройств.

В правом окне рис. 2.35 видны пиктограммы примеров. Они стали появляться еще в версиях пакета Simulink 6.*. Однако работа над примерами применения продолжается, и не все примеры даже в Simulink 7 выглядят полностью законченными. Некоторые до сих пор не имеют пиктограммы.

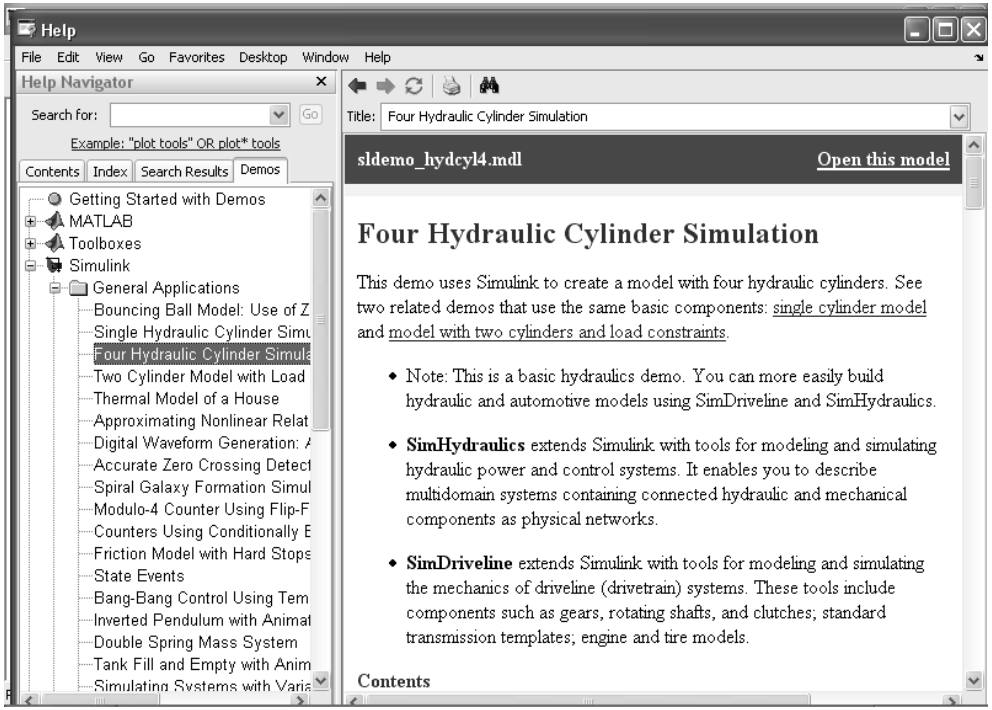


*Рис. 2.33. Окно справки MATLAB R2007b + Simulink 7 с открытой вкладкой **Demos***

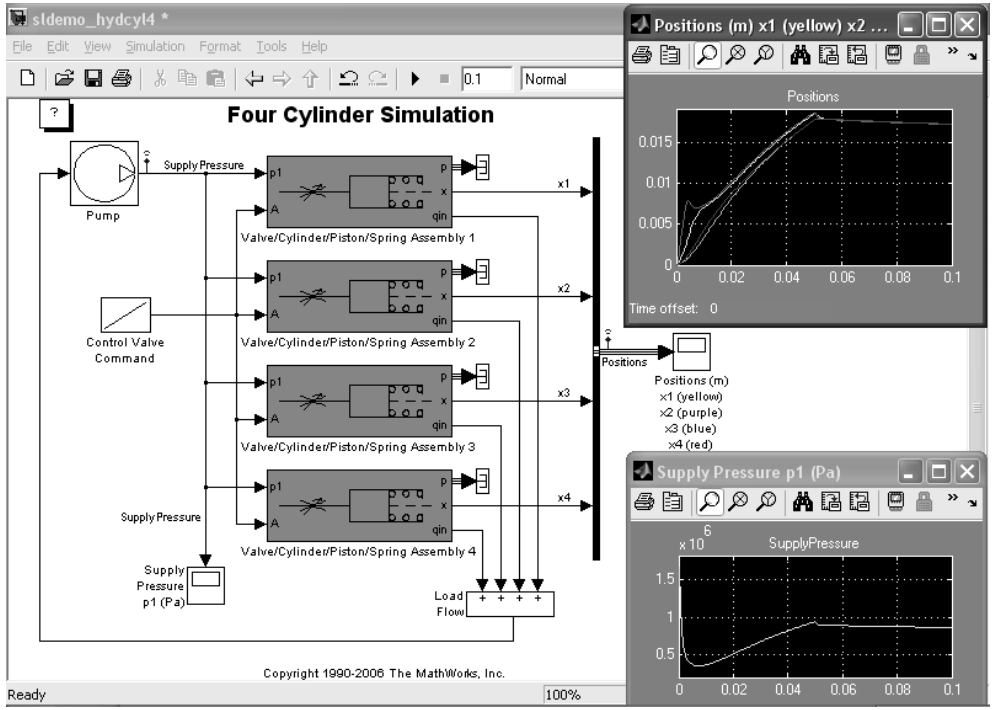


Рис. 2.34. Окно диаграммы и результаты моделирования для модели гидравлического устройства с четырьмя цилиндрами

Работа с примерами из пакетов расширения Simulink ничем не отличается от работы с примерами самой системы Simulink. Так, на рис. 2.36 показана работа с примером, иллюстрирующим заряд и разряд Ni-Cd аккумуляторной батареи. На этом рисунке представлена диаграмма модели, а также окно установки параметров блока Gain.

Уместно отметить, что возможности окон задания параметров ряда блоков в Simulink 7 несколько расширены. Это видно из изображения окна установки параметров блока Gain. Этот блок имеет три вкладки, с помощью которых возможна установка различных параметров.

Для отображения результатов моделирования наиболее часто применяется блок осциллографа Scope. Этот блок постоянно совершенствовался по мере выпуска новых реализаций пакета расширения Simulink. На рис. 2.37 показано окно данного блока для версии Simulink 7. Блок обеспечивает многоканальную регистрацию ряда сигналов.

В дальнейшем не стоит «зацикливаться» на мелких деталях интерфейса различных версий пакетов расширения систем MATLAB + Simulink. В задачи самоучителя не входит детальное отслеживание таких отличий. В дальнейшем материал книги изложен так, что эти отличия практически незаметны, а все сказанное в равной мере относится к любой версии системы MATLAB + Simulink.

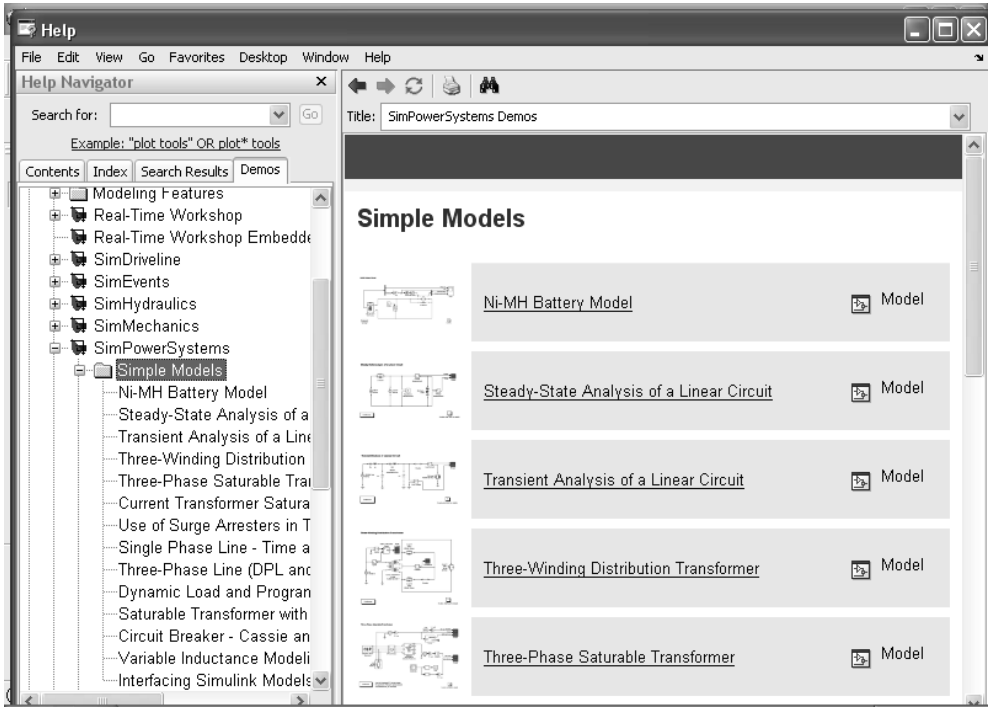


Рис. 2.35. Окно справки MATLAB R2007b + Simulink 7 по примерам пакета расширения SimPowerSystems

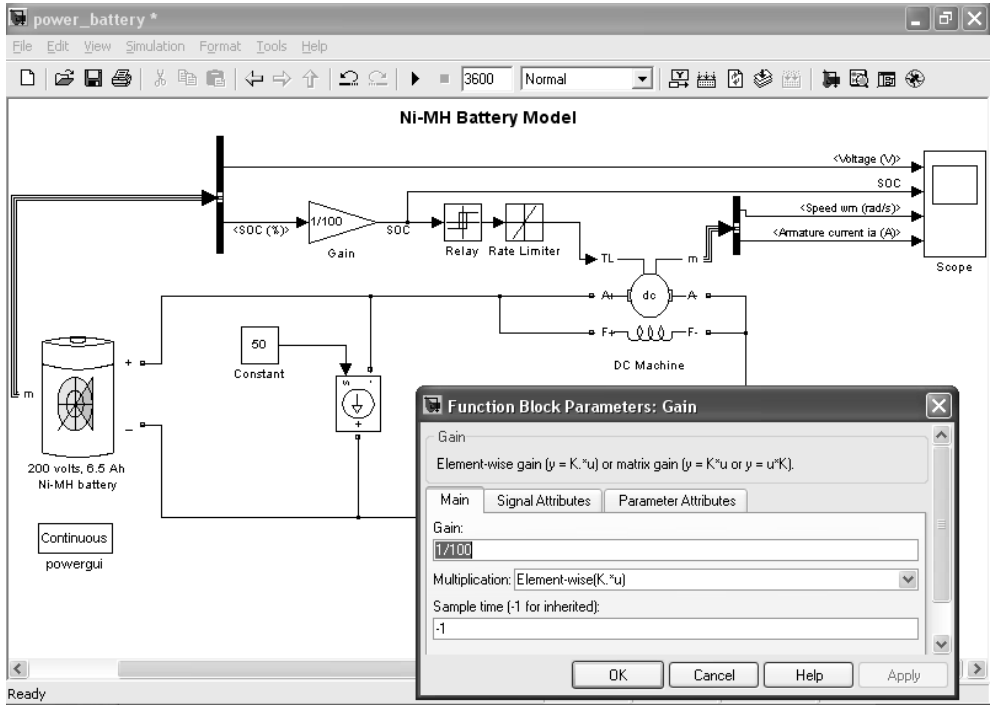


Рис. 2.36. Работа с примером моделирования заряда/разряда аккумуляторной батареи

Рис. 2.37. Окно виртуального осциллографа

Урок 3

Работа Simulink с файлами

Как и базовая система MATLAB, пакет расширения Simulink сохраняет диаграммы (модели) в виде файлов и имеет ряд библиотек с различными наборами блоков. В данном уроке описан доступ к файлам и разделам библиотек в пакете Simulink 5.*/6.*/7.

3.1. Интерфейс браузера библиотек

3.1.1. Окно браузера библиотек Simulink 5

Библиотеки Simulink представляют собой обширный набор поименованных блоков, имеющих графическое изображение (пиктограмму) и содержащихся в отдельных файлах. Для их просмотра служит браузер библиотек. На рис. 2.21 было показано окно браузера библиотек для новейшей версии Simulink 6.6, вошедшей в MATLAB R2007a. На рис. 3.1 это окно показано для версии Simulink 5.1 с выделенным разделом библиотеки. Блоки этого раздела видны в правой части окна. Окно имеет заголовок, меню, панель инструментов, поле информационных сообщений, окна с деревом библиотек и компонентами выделенного раздела библиотек и строку состояния (внизу окна).

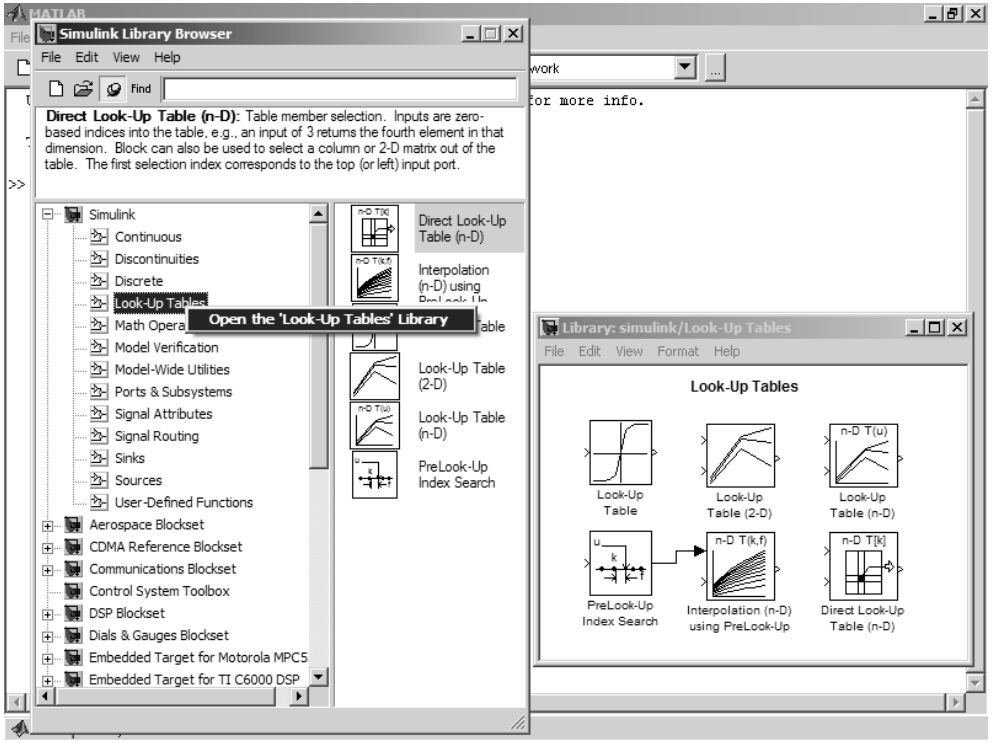


Рис. 3.1. Окно браузера библиотек Simulink 5.1 с открытым разделом Look-Up Table

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 ' Look-Up Table' Library** (Открыть библиотеку Simulink). Окно соответствующего раздела библиотеки Simulink появится на экране – на рис. 3.1 оно показано справа.

Как видно по рис. 3.1 (и рис. 2.21), 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.

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

3.1.2. Состав основной библиотеки блоков

Версия пакета Simulink 5 имеет существенно обновленную и расширенную (в сравнении с Simulink 4) 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 выше, для вывода специального окна с разделами основной библиотеки Simulink (рис. 3.1) 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. В Simulink 5 в этом окне содержатся следующие библиотеки:

- **Continuous** – компоненты с непрерывными характеристиками;
- **Discontinious** – компоненты с разрывными характеристиками;
- **Discrete** – дискретные компоненты;
- **LookUp Table** – табличное задание зависимостей;
- **Math Operations** – математические компоненты;
- **Model Verifications** – верификация моделей;
- **Model-Wide Utilities** – дополнительные утилиты;
- **Port & Subsystem** – порты и подсистемы;
- **Signal Attributes** – блоки атрибутов сигналов;
- **Signal Routing** – блоки маршрутизации сигналов;
- **Functions & Tables** – функции и таблицы;
- **Nonlinear** – нелинейные компоненты;
- **Connections** – соединительные компоненты;
- **Signals & Systems** – сигналы и системы;
- **Sinks** – регистрирующие устройства;
- **Sources** – источники сигналов и воздействий;
- **User-Defined Functions** – функции, задаваемые пользователем.

Постепенно мы рассмотрим состав каждого из упомянутых разделов основной библиотеки блоков. Уместно отметить, что состав разделов библиотеки Simulink 6 существенно отличается от состава разделов библиотеки Simulink 5. Эти отличия были отмечены в конце урока 2 и видны из сравнения рис. 2.21 и 3.1.

3.1.3. 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.

3.1.4. 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.

Позиция меню **File** имеет всего три команды:

- **New** – создание S-модели (команда **Model**) или библиотеки (команда **Library**);
- **Open…** – загрузка файлов S-модели и их поиск в файловой системе;
- **Preferences** – настройка Simulink.

Создание пользователем своей Simulink-модели (S-модели) мы рассмотрим далее в уроке 14. Команда **File** ⇒ **New** ⇒ **Library** позволяет создать библиотеку, в которую пользователь может поместить нужные ему компоненты. Действие команды **File** ⇒ **Open…** рассмотрено в уроке 12 на примере загрузки ряда демонстрационных примеров.

3.1.5. Настройка параметров Simulink

Команда **File** ⇒ **Preferences** выводит окно установки ряда параметров системы MATLAB + Simulink. Это окно для пакета Simulink 5.1 показано на рис. 3.2. Для пакета расширения Simulink 6.6 системы MATLAB R2007a окно описано в конце данного урока.

Здесь мы рассмотрим только те его возможности, которые относятся к пакету Simulink (окно **Preferences** есть и в базовой системе MATLAB). Окно **Preferences** с открытой ветвью дерева параметров Simulink и панелью установки шрифтов **Simulink Font Preferences** показано на рис. 3.3. 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. 3.4. Эта панель имеет четыре вкладки:

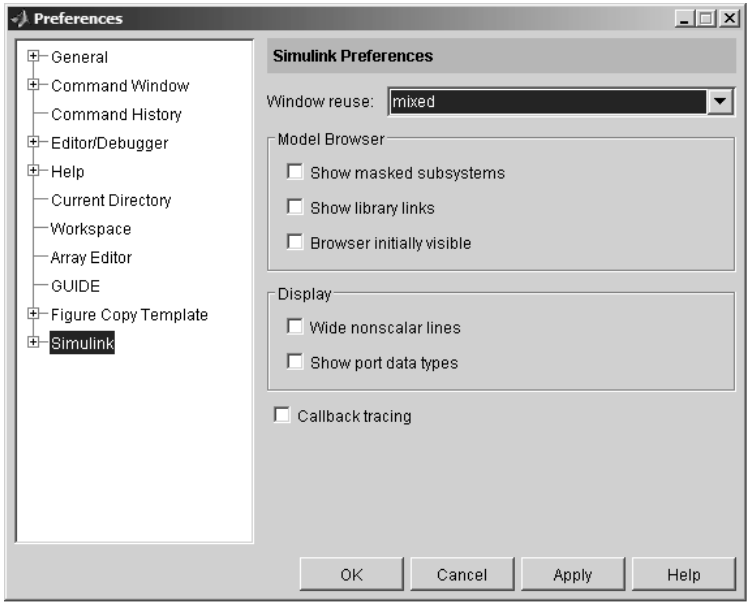


*Рис. 3.2. Окно **Preferences** Simulink 5.1 с разделом Simulink*

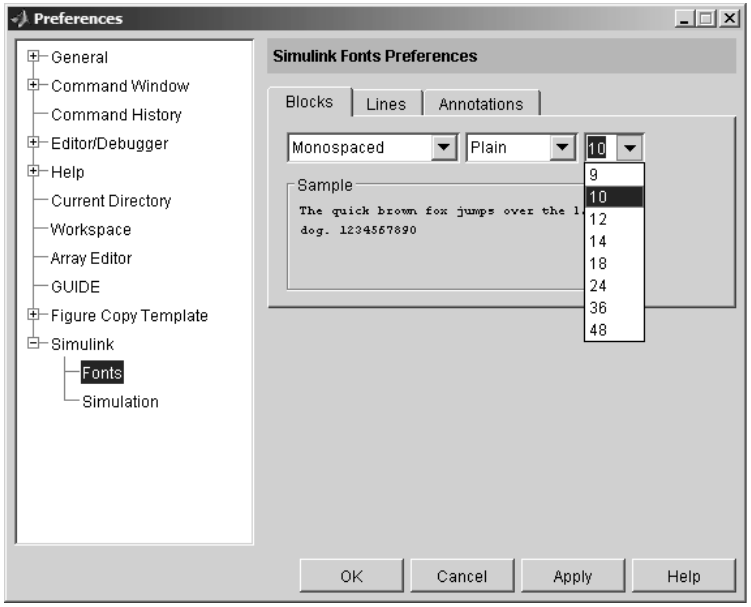


*Рис. 3.3. Окно **Preferences** Simulink 5.1 с установками шрифтов*

- **Solver** – установка параметров решателя дифференциальных уравнений;
- **Workspace** – установка параметров рабочего пространства;
- **Diagnostics** – установка параметров диагностики;
- **Advanced** – расширенные установки.

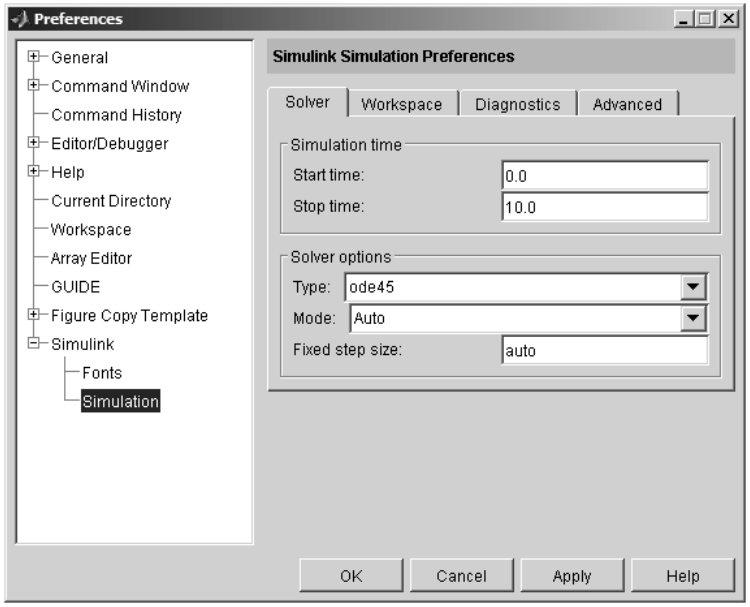


*Рис. 3.4. Окно **Preferences** Simulink 5.1 с установками параметров моделирования*

Наиболее важные параметры устанавливаются на вкладке **Solver**. Прежде всего это выбор начального и конечного условного времени моделирования и выбор типа решателя дифференциальных уравнений. Вкладка **Workspace** имеет всего две опции – записи времени и выхода в рабочее пространство. Они задают сохранение в рабочем пространстве времени и выхода модели. По умолчанию обе эти опции включены. Вкладка **Diagnostics** также имеет всего две опции, устанавливающие тип диагностических ошибок из двух списков.

Все эти вкладки были и в окне Simulink 4. В Simulink 5 появилась еще одна вкладка – **Advanced**. Она содержит две опции: **Block Reduction** (редукция блока) и **Boolean Logic signals** (сигналы с логикой Буля).

3.1.6. Меню Edit браузера библиотек

В меню **Edit** браузера библиотек имеются 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. 3.5).

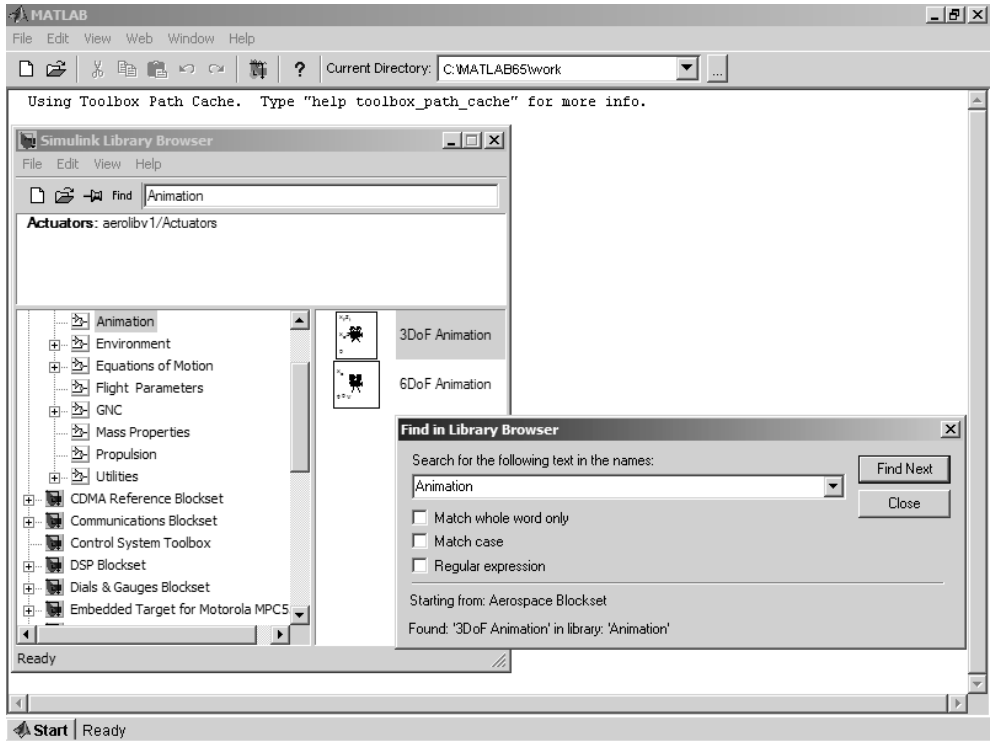


Рис. 3.5. Первый поиск блока по заданному имени

Команда **Find next block…** служит для поиска следующего блока, в который входит заданное слово (рис. 3.6). Эту же операцию выполняет и команда **Find next** в окне задания слова для поиска.

Если в результате повторного поиска больше блоков не обнаруживается, то выводится небольшое окно с сообщением об этом – оно также показано на рис. 3.5 над окном поиска.

3.1.7. Меню View браузера библиотек

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

- **Toolbar** – вывод/скрытие панели инструментов;

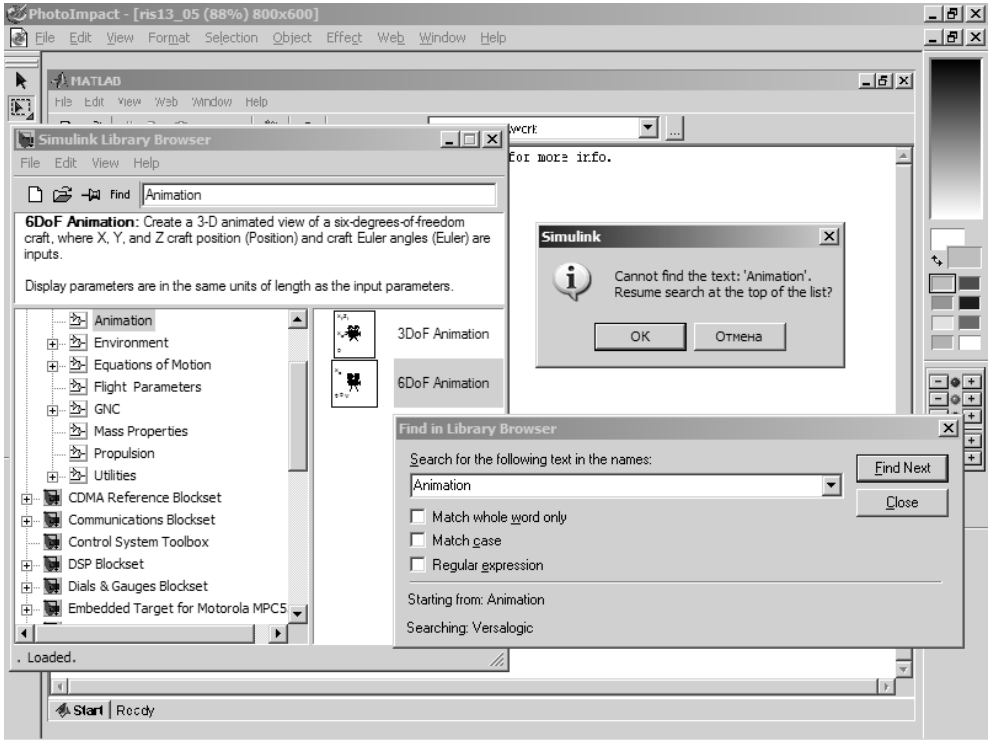


Рис. 3.6. Повторный поиск блока по заданному имени

- **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.

Действие всех этих команд вполне очевидно.

3.1.8. 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.

3.1.9. Панель инструментов окна браузера библиотек

Как видно по 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.

Все эти команды очевидны.

3.2. Интерфейс окна моделей Simulink

3.2.1. Панель инструментов окна моделей

Команда **New model** в окне браузера библиотек открывает пустое окно модели пакета Simulink. В нем можно сразу конструировать модель. Окно имеет панель инструментов.

Как видно по рис. 3.7, панель инструментов окна модели содержит 18 кнопок и список режимов работы. Номера их указаны на рис. 3.7. Назначение элементов панели инструментов следующее:

1. **New model** – создание новой модели;
2. **Open model** – загрузка модели;
3. **Save** – сохранение текущей модели;
4. **Print** – печать текущей модели;
5. **Cut** – перенос выделенного объекта в буфер;
6. **Copy** – копирование выделенного объекта в буфер;
7. **Paste** – вставка объекта из буфера;
8. **Undo move** – отмена последней операции;
9. **Redo** – восстановление последней отмененной операции;
10. **Start simulation** – запуск моделирования;
11. **Stop simulation** – остановка моделирования.
12. **Normal/Accelerator/External** – список режимов работы;

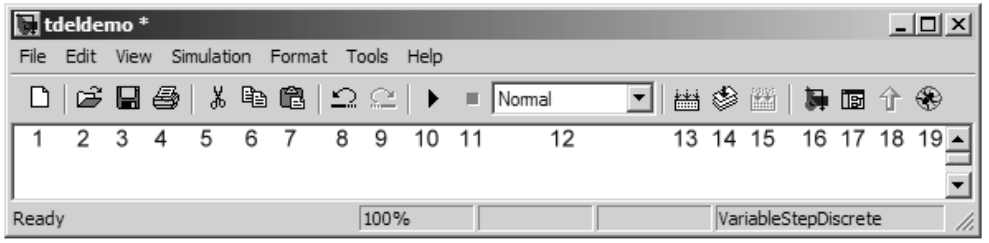


Рис. 3.7. Окно S-модели пакета Simulink

13. **Build All** – создание исполняемого кода модели с помощью Real Time Workshop;
14. **Update diagram** – обновление окна модели;
15. **Build subsystem** – создание исполняемого кода подсистемы.
16. **Library Browser** – открытие браузера библиотек;
17. **Toggle model browser** – открытие браузера моделей в левой части окна модели;
18. **Go to parent system** – переход в основную (родительскую) систему;
19. **Debug** – переход в режим отладки модели.

3.2.2. 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.

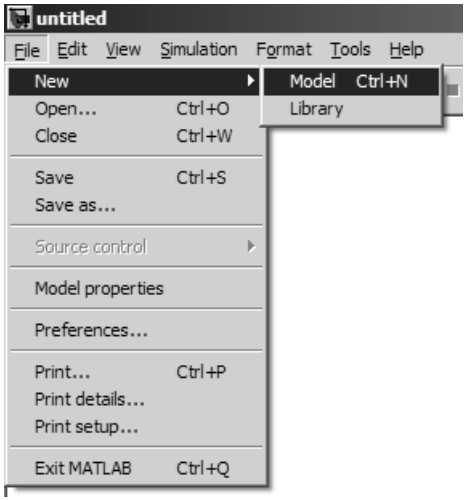


*Рис. 3.8. Меню окна модели Simulink с открытой позицией **File***

3.2.3. Меню File окна модели

Меню **File** содержит ряд команд для работы с файлами (рис. 3.8):

- **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.

3.2.4. Контроль источников

Команда **Source Control…** по умолчанию отключена. Она включается на вкладке **General** окна **Preferences**. Если она возможна, то 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, которые показаны на рис. 3.9. В окне **Check out** предусмотрена опция **Keep checked out** – пропускать контроль выхода. По умолчанию она не задействована. В окне **Check in** имеются следующие опции: **Latest** (использование последней модели источника), **Lock** (закрытие от модификации модели) и **Specific** (задание спецификации в специальном поле справа от этой опции). Первые две опции по умолчанию включены, а последняя отключена.

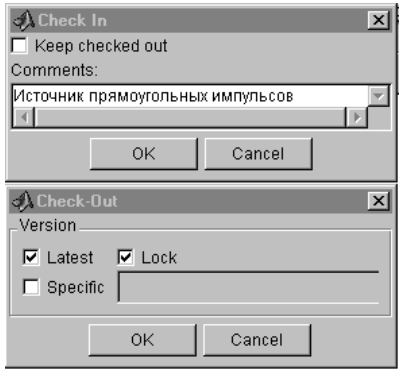


Рис. 3.9. Окна контроля источников

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.

3.2.5. Вывод окна свойств текущей модели

Команда **Model Preferences** выводит окно свойств текущей модели (рис. 3.10). Это окно имеет три вкладки: **Summary**, **Callback** и **Histoty**. Вкладка **Summary** 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).

Вкладка **Callback** позволяет задать ряд параметров, которые, как правило, отсутствуют или используются по умолчанию. Назначение параметров можно понять по рис. 3.11. Задаются функции загрузки и инициализации модели, пуска и остановки, записи и закрытия.

Вкладка **History** (рис. 3.12) несет данные о модификации системы – если, разумеется, таковые есть. Кнопка **Edit** открывает доступ к окну сохранения модификаций модели (рис. 3.12).

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

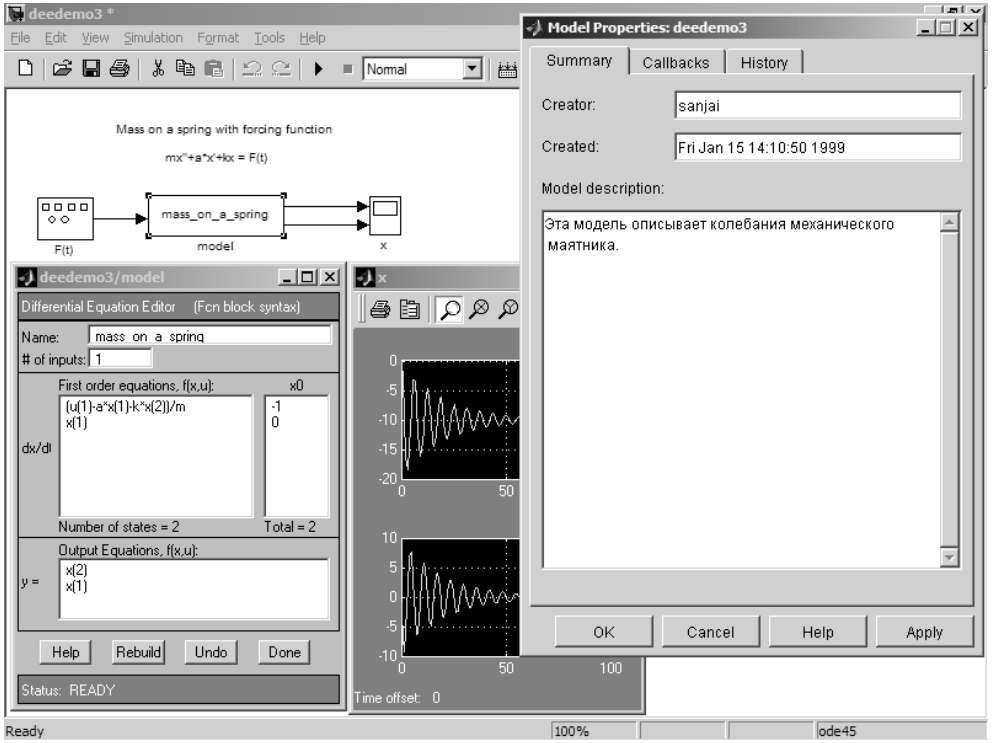


Рис. 3.10. Модель с окном ее свойств

3.3. Печать текущей модели

3.3.1. Вывод окна печати модели

Команда **File ⇒ Print...** выводит окно печати модели. Это окно (см. рис. 3.13) 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/XP 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).

3.3.2. Настройка принтера

Кнопка **Properties** в правом верхнем углу окна печати выводит окно свойств выбранного принтера. На рис. 3.14 представлено окно свойств принтера для цветного струйного принтера/копира/сканера HP PSC 1100 фирмы Hewlett Packard, ко-

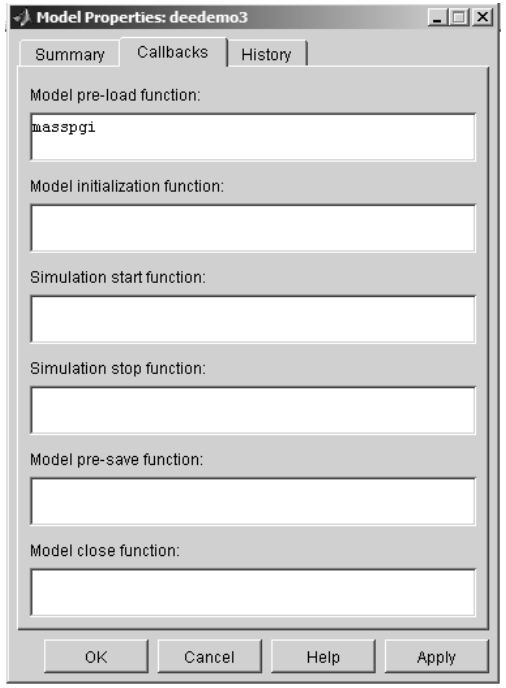


*Рис. 3.11. Вкладка **Options** окна свойств модели*

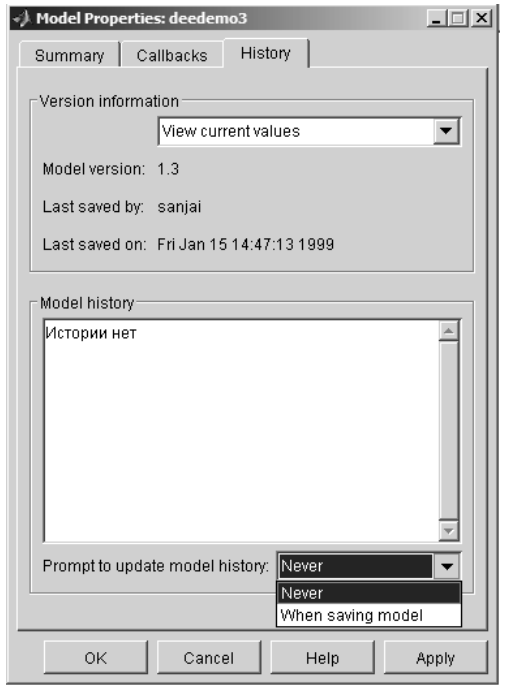


*Рис. 3.12. Окно свойств модели с открытой вкладкой **History***

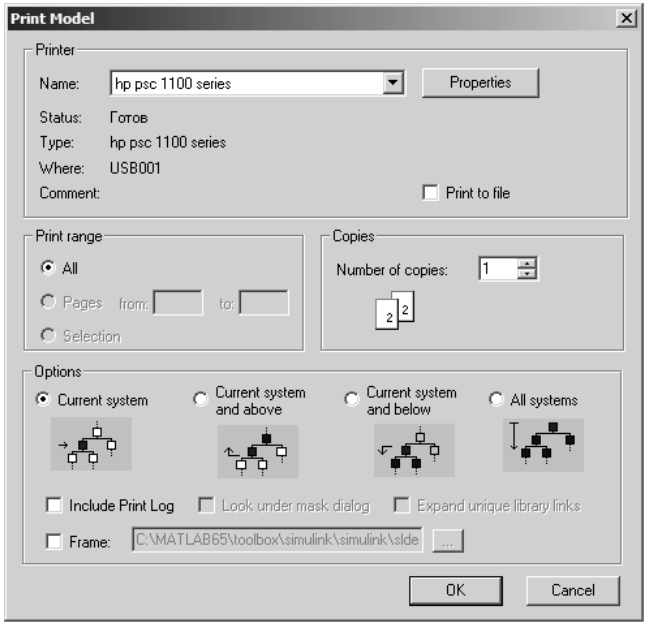


Рис. 3.13. Окно печати текущей модели

торый используется автором. Это стандартное окно Windows, и его вид зависит от драйвера заданного принтера. Для указанного принтера окно содержит две вкладки для установки качества печати, типа бумаги и ориентации печати (вкладка **Настройка**) и запуска утилит профилактики принтера (вкладка **Службы**).

Команда **Print Details…** выводит окно, показанное на рис. 3.15. В нем можно установить директорию, файл из которой печатается, а также задать печать текущего объекта отдельно или вместе с ближайшими связанными объектами. Характер связи объектов представлен снизу окна.

Команда **Print Setup…** в позиции **File** меню окна Simulink открывает окно настройки принтера (рис. 3.16). 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. 3.14).

3.4. Особенности Simulink 6/7

3.4.1. Новое окно Preferences Simulink 6/7

Приведенный выше материал описания Simulink 5 полностью соответствует и новым версиям Simulink 6.*. В частности, идентичны возможности позиции меню **File**. Однако некоторые видимые отличия есть. Например, окно **Preferences**

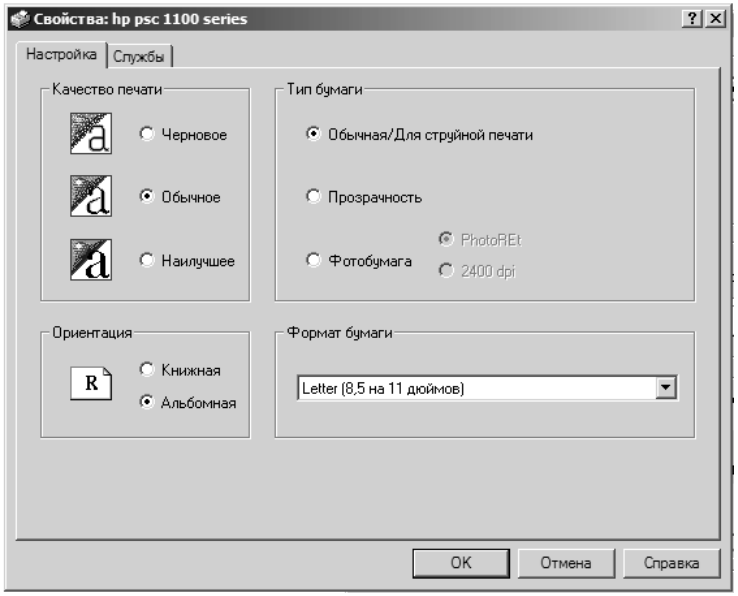


Рис. 3.14. Окно настройки принтера

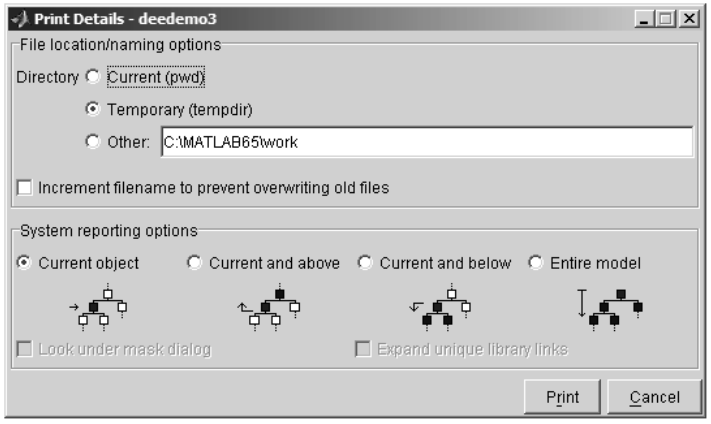


*Рис. 3.15. Окно **Print Details***

в Simulink 6.* (и в соответствующих версиях базовой системы MATLAB) модернизировано. На рис. 3.17 это окно показано для версии Simukink 6.6.

Как и для Simulink 5.1, ниже мы рассмотрим только те возможности окна **Preferences**, которые относятся к пакету Simulink. Окно **Preferences** с открытой ветвью дерева параметров Simulink и панелью установки шрифтов **Simulink Font Preferences** показано на рис. 3.18. Установки в этом окне вполне очевидны – они относятся к выбору шрифтов для различных компонентов библиотек Simulink.

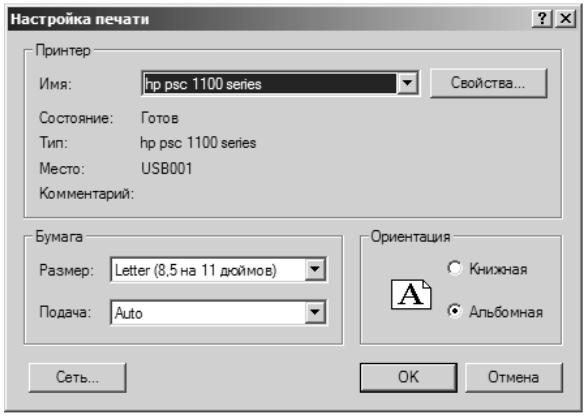


Рис. 3.16. Окно установки принтера

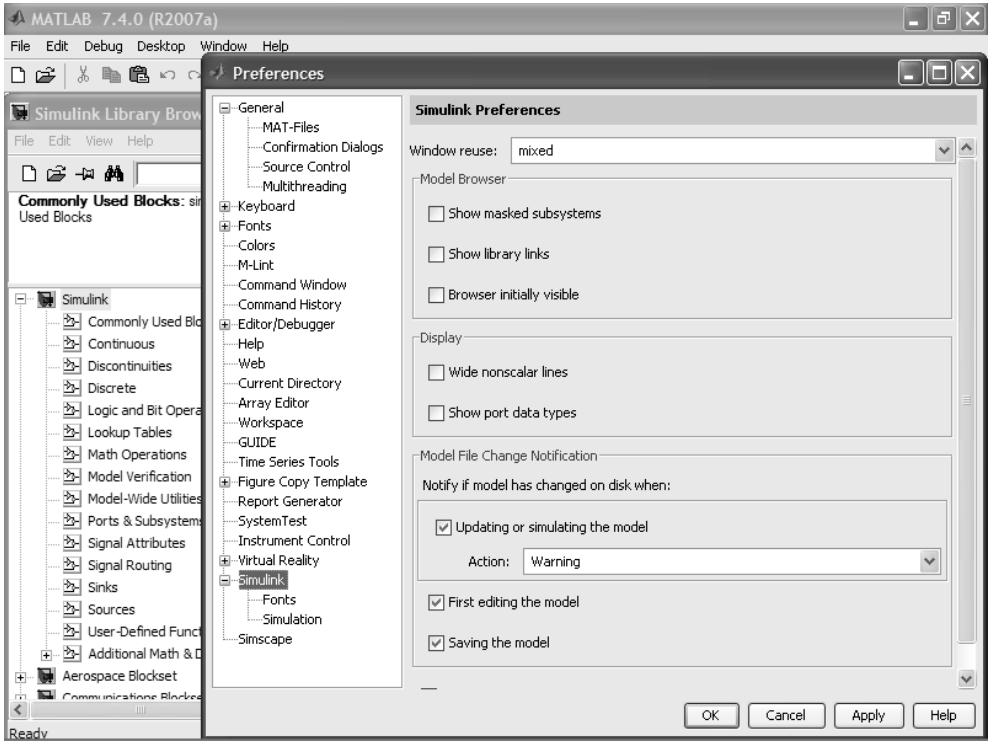


*Рис. 3.17. Окно **Preferences** Simulink 6.6 с разделом Simulink*

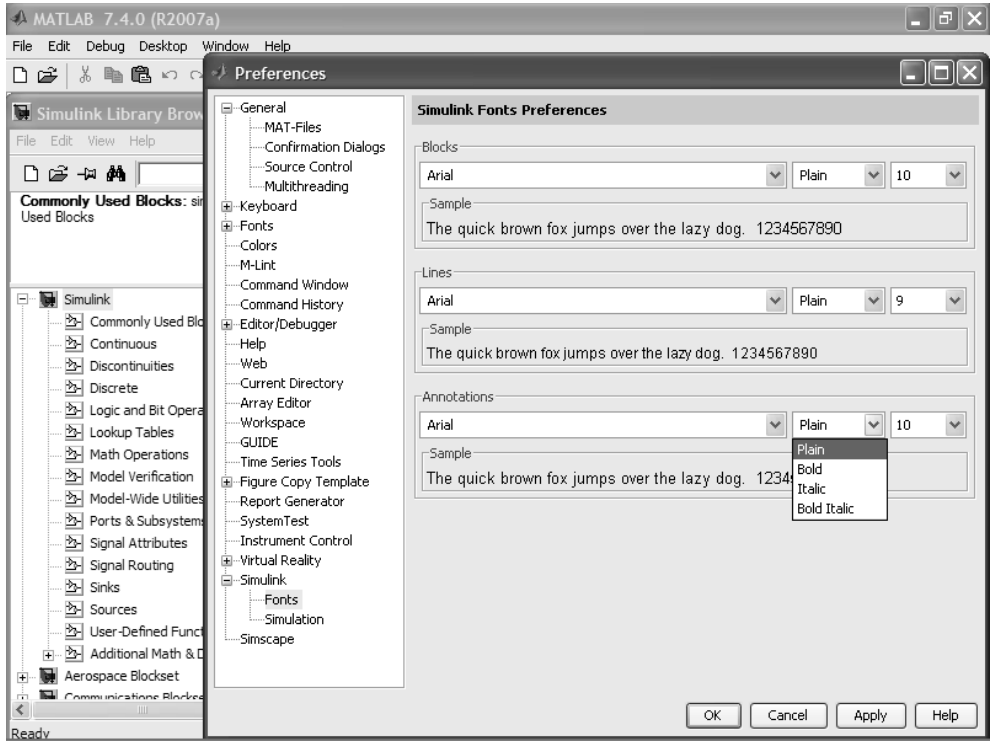


*Рис. 3.18. Окно **Preferences** Simulink 6.6 с установками шрифтов*

То же окно, но с панелью установок параметров моделирования Simulink **Simulation Preferences** показано на рис. 3.19. Содержимое этой панели резко отличается от содержимого соответствующей панели Simulink 5.1, показанной на рис. 3.4. На панели Simulink 6.6 нет никаких вкладок, но появилась кнопка **Launch Model Explorer (Запуск обозревателя модели)**.

3.4.2. Новое окно обозревателя модели Simulink 6/7

Активизация кнопки приводит к выводу окна обозревателя текущей модели. Это окно внушительного вида (рис. 3.20) открывает доступ ко всем установкам и параметрам текущей модели. Так, на рис. 3.20 представлены установки решателя Solver.

Вряд ли начинающий пользователь будет серьезно работать с окном обозревателя модели. Все установки его доступны и с помощью других, более простых средств, которые используются в ходе подготовки модели и ее запуска. Однако для опытного пользователя обозреватель модели представляет уникальное сред-

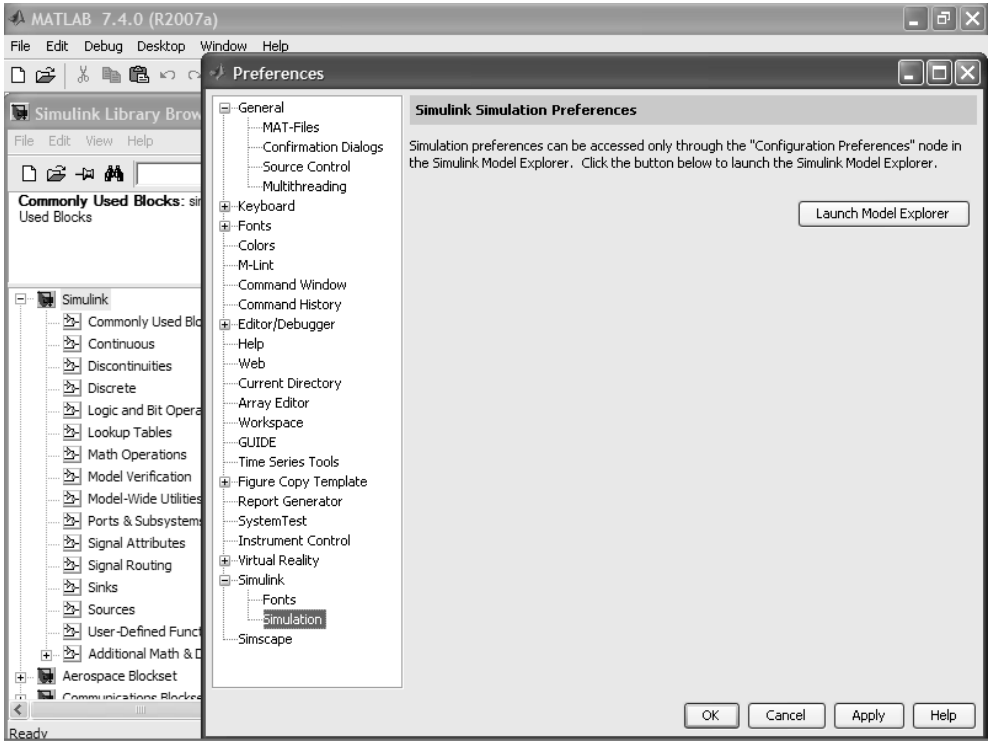


*Рис. 3.19. Окно **Preferences** Simulink 6.6 с панелью **Simulations***

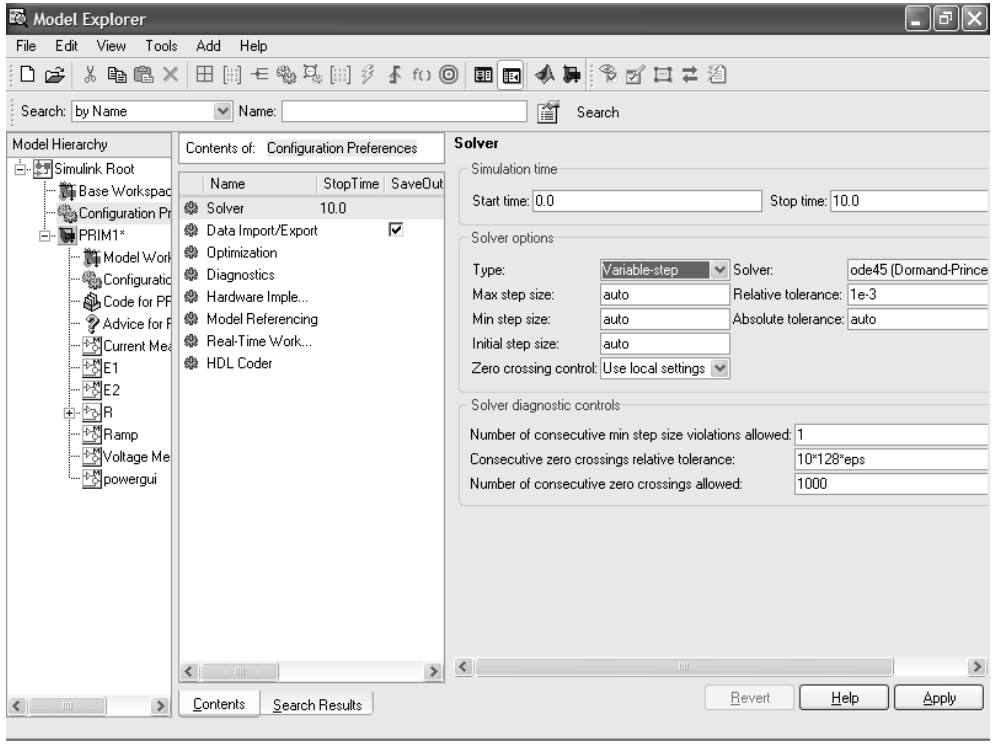


Рис. 3.20. Окно обозревателя модели Simulink 6.6

ство оценки и изменения практически всех параметров модели и ее отдельных блоков. Для опытного пользователя не составит особого труда изучить возможности этого средства.

3.4.3. Окно модели Simulink 6/7 и контекстное меню

Детали интерфейса окна модели в Simulink 6.6 (рис. 3.21) практически не отличаются от таковых для Simulink 5.1. В окне рис. 3.21 представлена модель, описывающая зависимость мощности в нагрузке с переменным сопротивлением, подключенная к двум источникам напряжения со своими внутренними сопротивлениями, включенными последовательно. Мощность вычисляется как произведение напряжения на нагрузке на ток в ней.

В процессе создания модели (он описан в следующем уроке и одинаков для любой версии Simulink) надо ввести все блоки и установить их параметры. Заметим, что удобным средством контроля всех параметров и характеристик блоков является контекстно-зависимое меню правой клавиши мыши. Оно вводится нажатием левой клавиши мыши после выделения блока (наведением на него курсора мыши и нажатием левой клавиши мыши). На рис. 3.21 показано такое меню для блока осциллографа Scope 2. Разумеется, контекстное меню можно использовать и в более ранних версиях Simulink.

3.4.4. Пуск модели в Simulink 6/7

Пуск модели в Simulink 6.6 производится аналогично тому, как он делается в предшествующих версиях. Перед пуском рекомендуется установить параметры моделирования. Это возможно разными способами. Можно, например, установить параметры решателя Solver в окне предпочтений **Preferences** – см. рис. 3.20. Это также возможно с помощью окна конфигурации модели, которое вызывается командой **Configuration Parameters…** в позиции **Simulation** меню. Это окно показано на рис. 3.22. Заметим, что данная модель рассчитана на применение фиксированного шага решения, параметры которого и представлены в окне установки параметров данной модели.

Пуск модели обычно производится активизацией кнопки **Start Simulation** со знаком *>* в панели инструментов Simulink 6.6 или командой **Start** в позиции

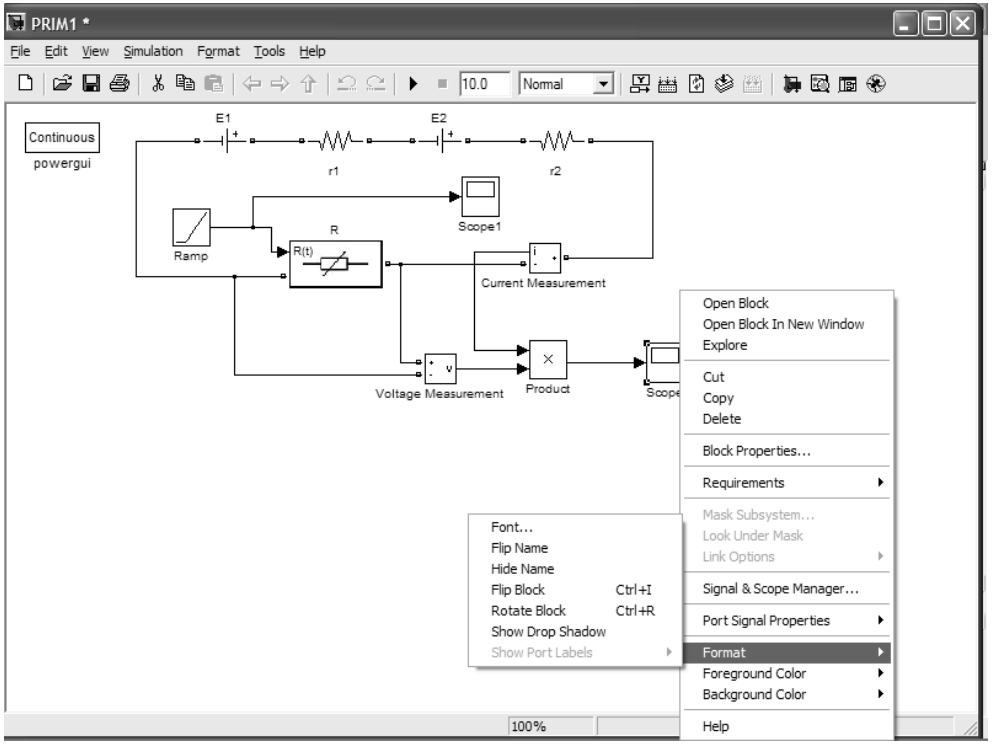


Рис. 3.21. Окно модели Simulink 6.6

Simulation меню. На рис. 3.23 показан результат пуска приведенной в качестве примера модели. Осциллографы фиксируют линейный закон нарастания сопротивления нагрузки во времени и рассеиваемой на ней мощности. Хорошо видно, что кривая мощности имеет пик, при котором сопротивление нагрузки равно сумме внутренних сопротивлений источников напряжений.

Справа от диаграммы модели представлен субблок резистора с переменным сопротивлением. Такого блока в составе стандартных библиотек Simulink нет. Подготовка подобных блоков и подсистем описана в уроке 8.

Как уже отмечалось (см. урок 2), новейшая реализация Simulink 7 имеет интерфейс, аналогичный описанному для Simulink 6.6.

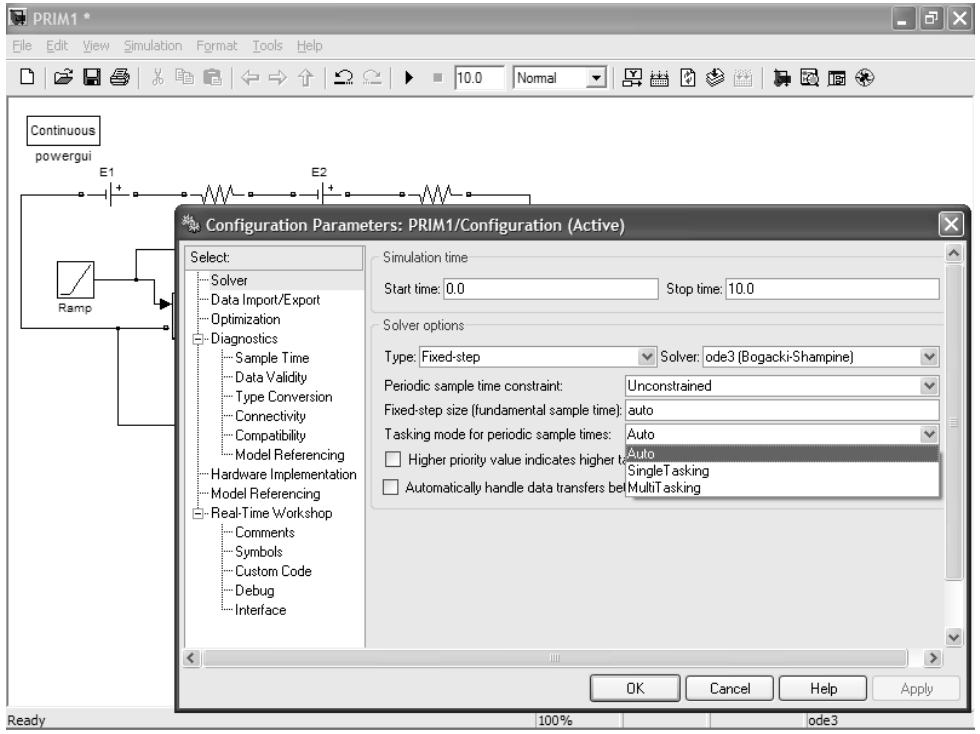


Рис. 3.22. Окно установки параметров модели в окне модели Simulink 6.6

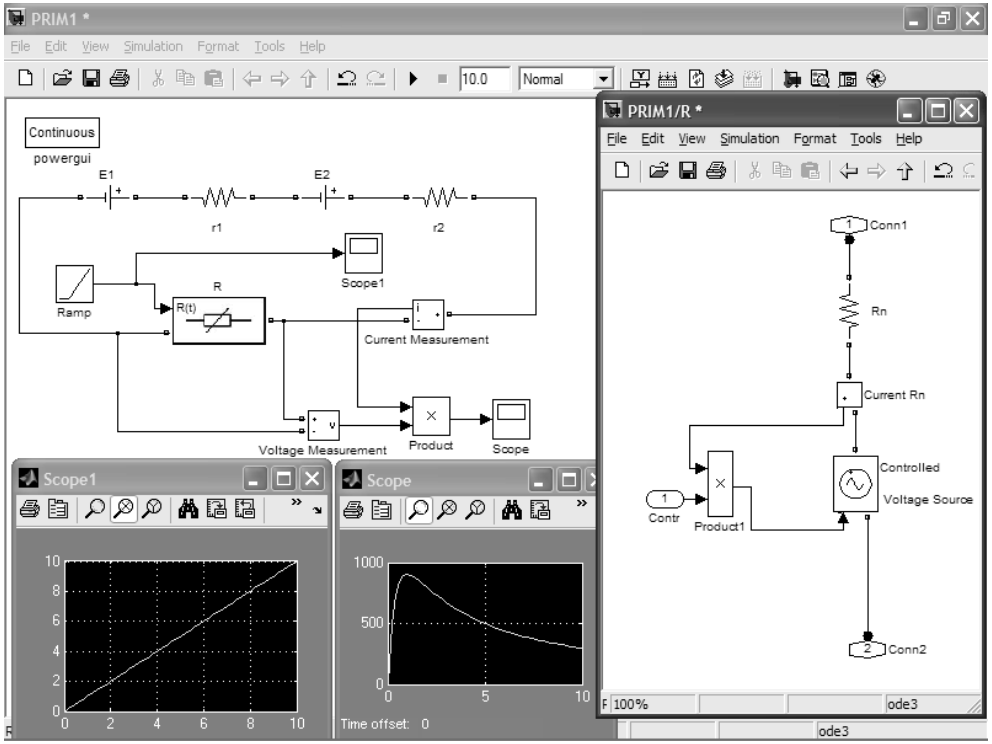


Рис. 3.23. Окно установки параметров модели в окне модели Simulink 6.6

Урок 4

Подготовка и запуск модели

В данном уроке описана подготовка диаграммы (модели) из отдельных блоков, входящих в различные библиотеки пакета расширения Simulink. Описаны способы соединений блоков, настройки их параметров и параметров запуска создаваемых моделей.

4.1. Создание модели

4.1.1. Постановка задачи и начало создания модели

Решение любой задачи в системе Simulink должно начинаться с ее постановки и оценки методов решения (см. урок 1). Чем глубже продумана постановка задачи, тем больше вероятность успешного ее решения в заданные сроки. 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 или команда **File** ⇒ **New** ⇒ **Model** в меню).

4.1.2. 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.

Тут уместно отметить, что хотя система MATLAB + Simulink не имеет локализованной под русский язык версии, надписи на русском языке можно вводить. Однако не всегда используемые по умолчанию шрифты поддерживают символы кириллицы.

4.1.3. Размещение блоков в окне модели

Из раздела библиотеки **Sources** перетащим мышью три источника сигнала: синусоидального, прямоугольного (дискретного) и пилообразного. Рисунок 4.1 показывает момент ввода третьего источника – генератора пилообразных импульсов.

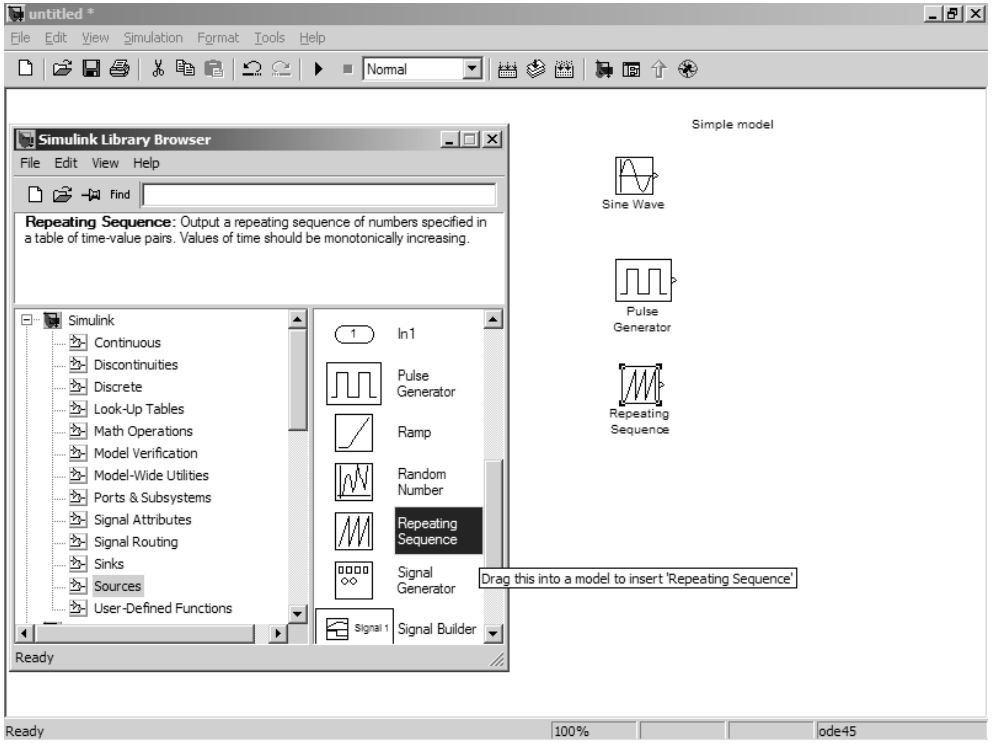


Рис. 4.1. Начало подготовки модели – ввод трех источников сигналов

Аналогично были введены два других источника – синусоидальных сигналов и прямоугольных импульсов (меандра).

Затем из раздела **Sinks** перетащим в окно модели блок осциллографа. Этот момент иллюстрируется рис. 4.2.

4.1.4. Выделение блока модели

Нам надо ввести еще два блока осциллографа. Их можно ввести, как уже описывалось. Но мы проделаем это иначе – используя буфер промежуточного хранения операционной системы Windows.

Обратите внимание, что только что введенный блок выделяется – выделение отмечается маленькими черными прямоугольниками по углам блока – рис. 4.2. На рис. 4.3 показано меню редактирования **Edit** в открытом виде – при выделении блока в этом меню становятся доступны команды редактирования свойств блока. В общем случае 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. 4.3 выделен блок осциллографа **Scope**.

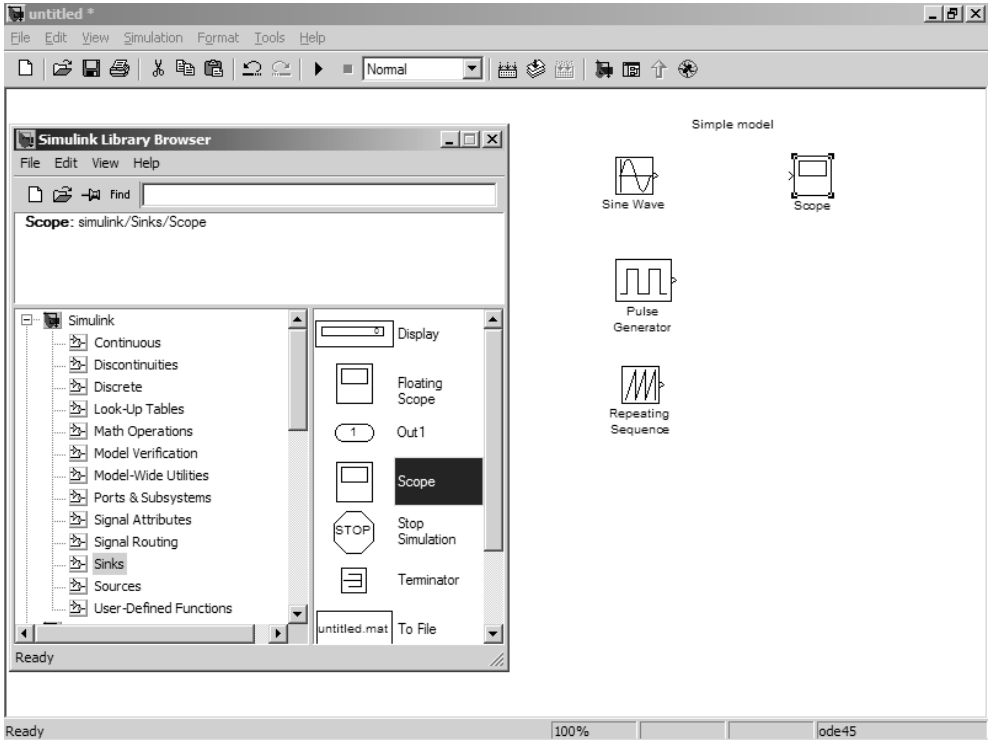


Рис. 4.2. Ввод блока осциллографа

Если захватить курсором мыши уголок выделенного блока, то можно заметить, что курсор мыши превратится в перекрестие тонких диагональных двухсторонних стрелок. Это означает, что можно пропорционально увеличивать или уменьшать блок в диагональных направлениях. Кроме того, выделенный блок можно перетаскивать мышью.

Следует отметить еще один способ выделения не только отдельных блоков, но и их группы. При нем надо установить курсор мыши в стороне от группы блоков и, нажав левую клавишу мыши и удерживая ее, начать перемещать ее. Появится прямоугольник из черных пунктирных линий, и все блоки, попавшие в него, выделяются. Позже мы рассмотрим пример такого выделения.

4.1.5. Меню редактирования Edit

Кратко рассмотрим основные команды меню **Edit** (рис. 4.3). Это меню содержит ряд типовых команд, которые разбиты на шесть групп. В первой группе есть две команды: **Undo** (отмена последней операции) и **Redo** (восстановление последней отмененной операции). Эти команды являются контекстно-зависимыми.

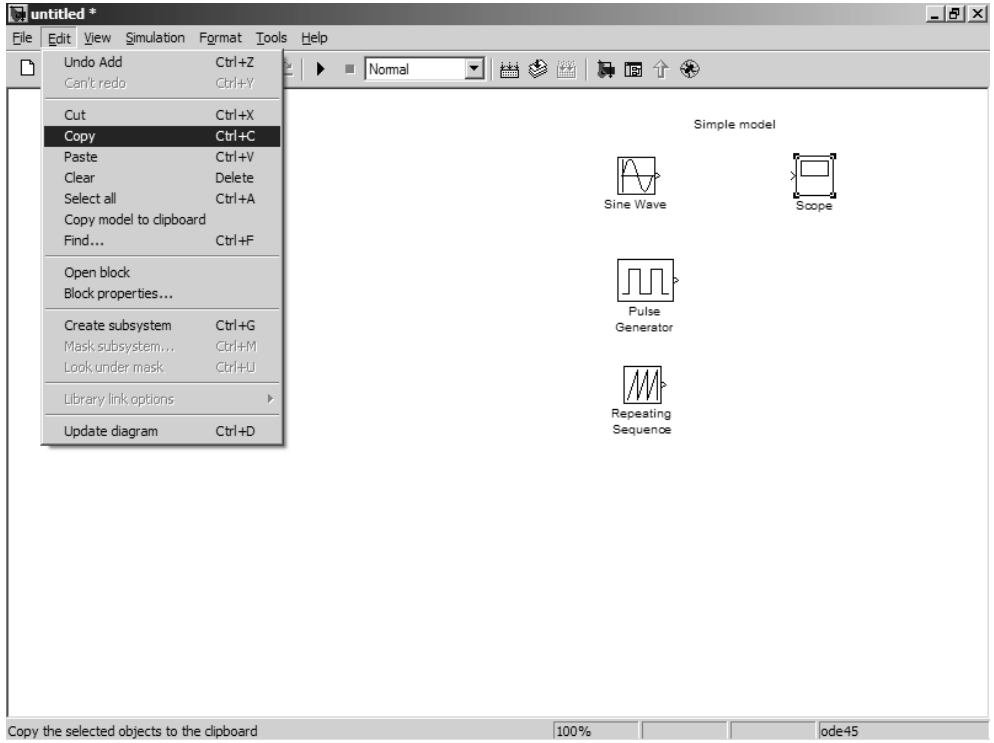


Рис. 4.3. Применение команды Copy помещения блока в буфер

Следующая группа команд связана 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.

4.1.6. Применение буфера обмена

Вернемся к нашей модели и покажем некоторые приемы работы с буфером обмена. На рис. 4.3 выделен блока 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. 4.4).

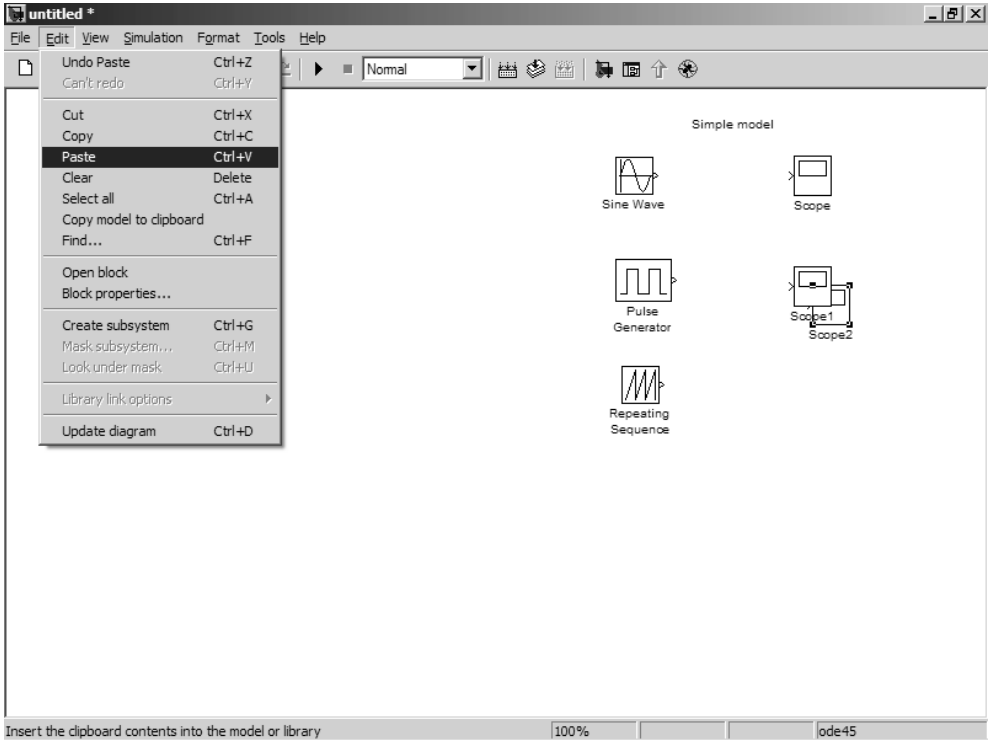


*Рис. 4.4. Размножение блока **Scope** с помощью буфера обмена*

Аналогично добавляется еще один блок – **Scope2**. После повторного использования команды **Paste** он виден наложенным на блок **Scope1**. Его можно перетащить вниз мышью. Теперь можно приступить к соединению выходов источников с 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) – рис. 4.5.

При первом соединении появляется окно, поясняющее способ облегчения соединения с помощью клавиши **Ctrl**. Завершив все соединения, получим модель, показанную на рис. 4.6.

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.*

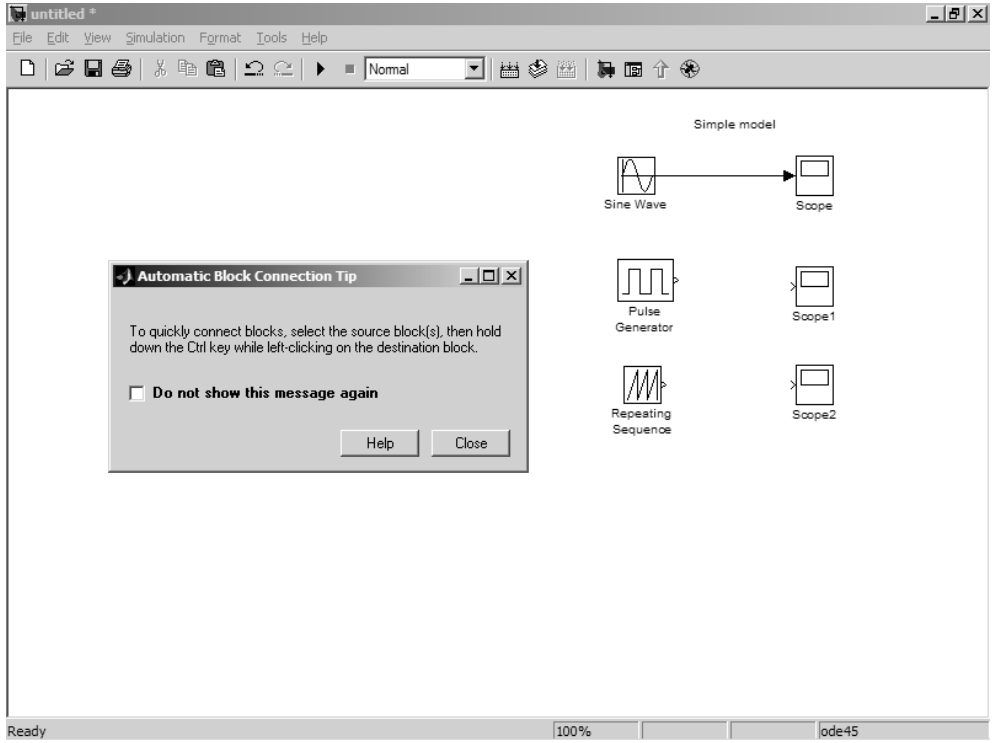


Рис. 4.5. Начало соединений между блоками модели

4.1.7. Выделение ряда блоков и их перенос

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

В первом способе для выделения надо использовать команду **Select all**. Этот способ иллюстрирует рис. 4.6. Во втором используется мышь. В стороне от выделяемых блоков необходимо установить курсор мыши и нажать левую клавишу. Теперь при перемещении мыши появится расширяемый прямоугольник из тонких пунктирных линий. Как только в нем окажется какой-то блок, он будет выделен. Охватив прямоугольником все блок, можно выделить их.

Выделенный набор блоков можно перетаскивать мышью как единый объект. Это показано на рис. 4.7, где справа виден набор исходных блоков, а слева – перетаскиваемый блок. Отпустив левую клавишу мыши, можно увидеть блоки на новом месте.

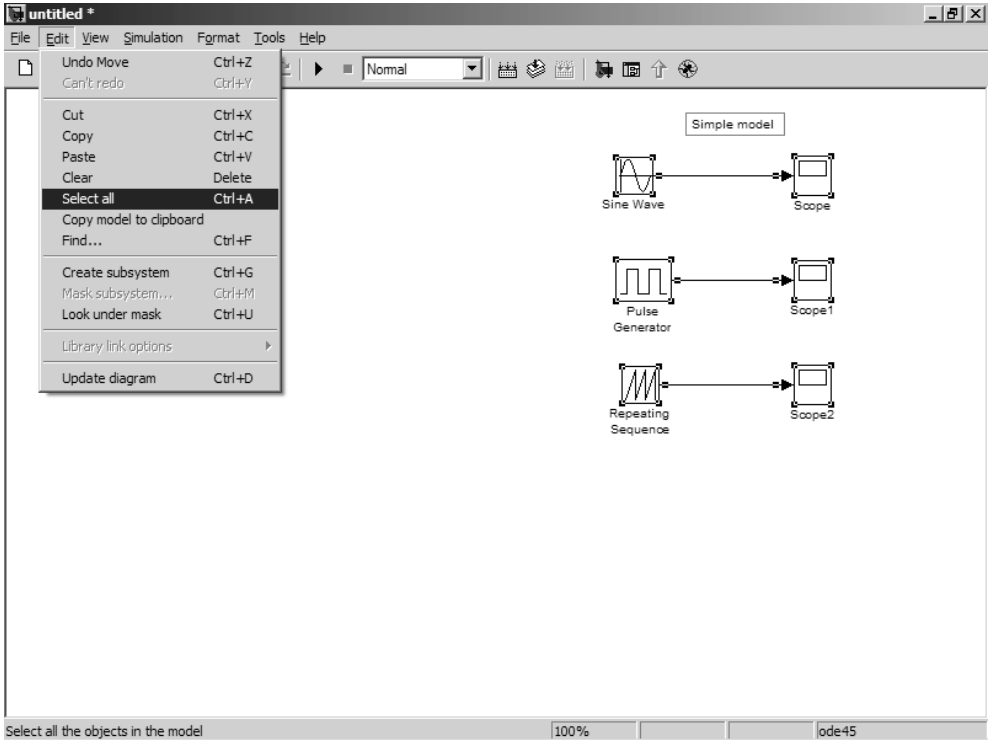


Рис. 4.6. Готовая модель

4.1.8. 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**=10 в окне установки параметров моделирования (напоминаем, что оно вызывается командой **Simulation ⇒ Simulation parameters...**). Этот момент представлен на рис. 4.8.

После этого, запустив моделирование нажатием кнопки **Start Simulation** с изображением треугольника или командой меню **Simulation ⇒ Start**, можно увидеть результат, показанный на рис. 4.9. 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 удобное для обзора положение. Их можно также растянуть или сжать

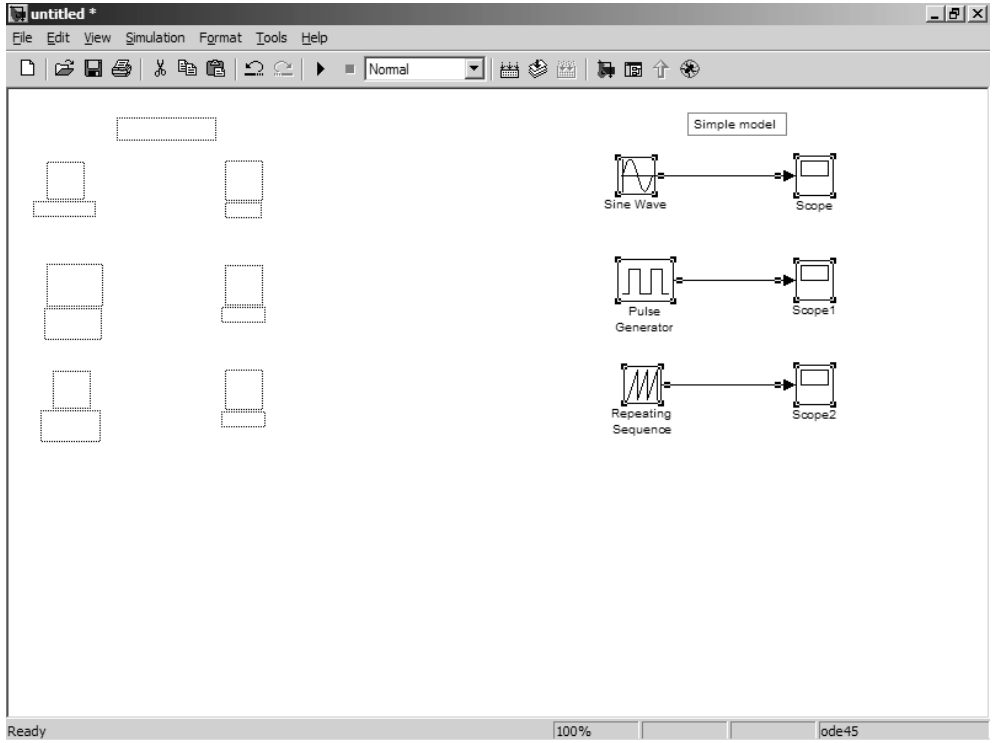


Рис. 4.7. Перенос группы блоков на новое место

в любом направлении с помощью мыши. В результате мы и получим вид экрана, представленный на рис. 4.9.

Итак, мы видим, что все три модели работают, и осциллограммы представляют временные зависимости сигналов, которые вырабатывают источники – синусоиду, прямоугольные импульсы и треугольные импульсы.

4.2. Моделирование ограничителя

4.2.1. Постановка задачи

В качестве следующего примера рассмотрим задачу моделирования работы идеального ограничителя сигналов, 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.

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 регистрирую-

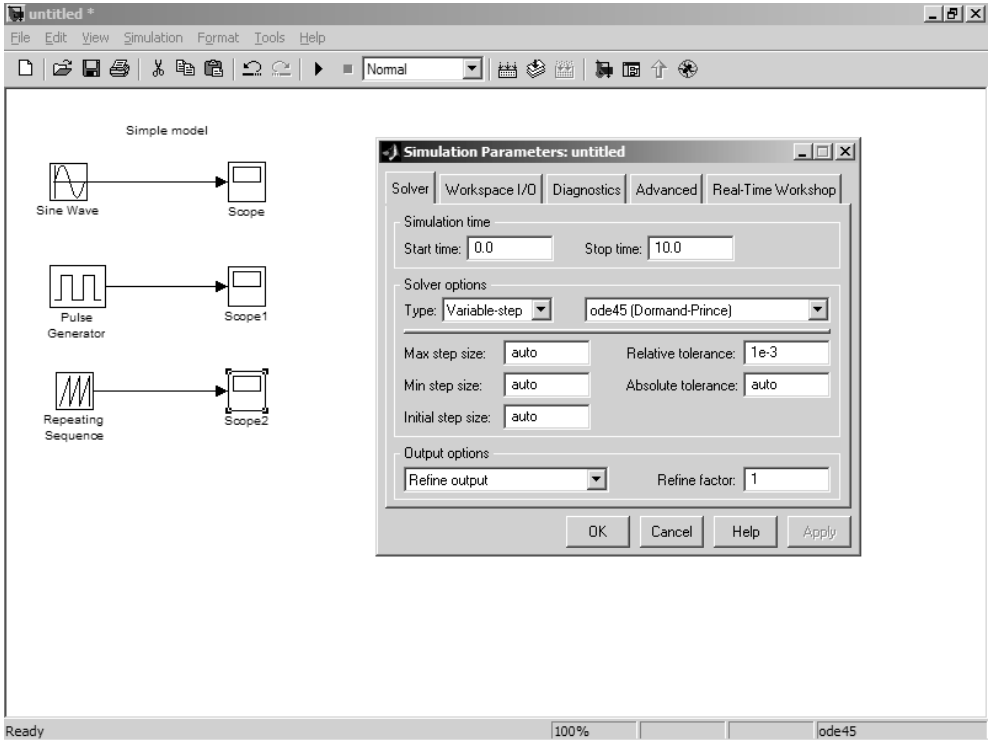


Рис. 4.8. Готовая модель и окно установки параметров моделирования

щий блок – осциллограф. Так как функциональная схема моделируемого устройства в данном случае вполне очевидна, то мы можем перейти к ее реализации.

4.2.2. Создание и запуск 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).

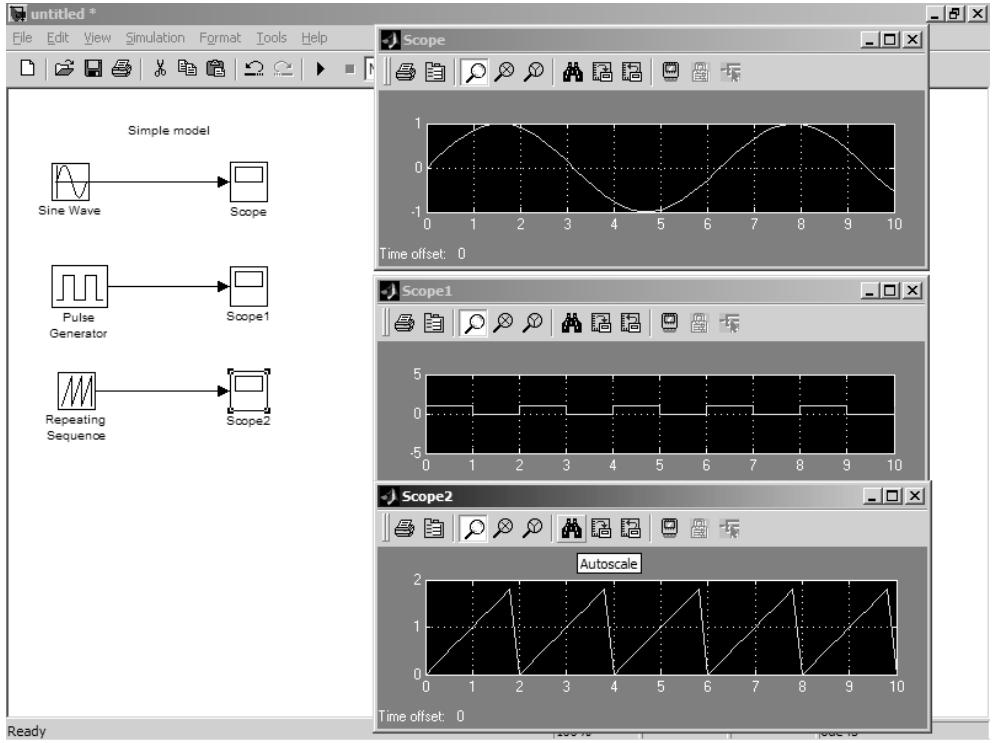


Рис. 4.9. Готовая модель и результаты ее моделирования

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. 4.10. Работа созданной модели (диаграммы) вполне очевидна и соответствует нашим представлениям о работе двухстороннего ограничителя.

4.2.3. Настройка масштаба осциллограмм

Нетрудно заметить, что масштаб отображения осциллограммы у осциллографа на рис. 4.10 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.

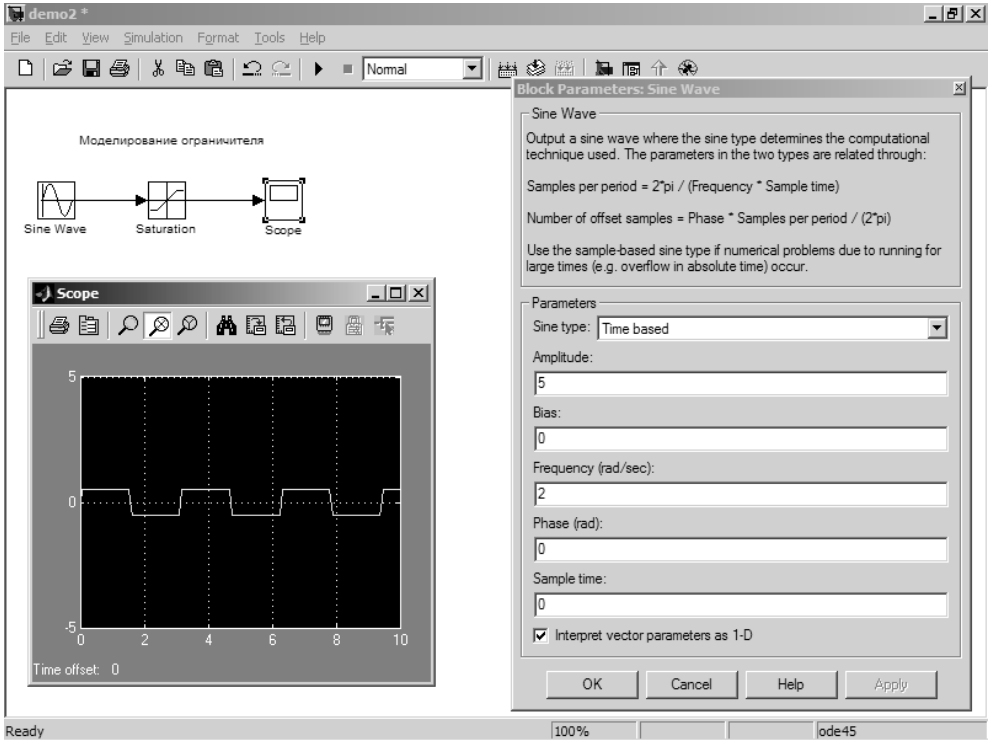


Рис. 4.10. Модель ограничителя и результат ее работы

Для смены масштаба достаточно щелкнуть правой кнопкой мыши в окне осциллограммы. В появившемся контекстном меню (рис. 4.11) нужно выбрать команду **Axes Properties…**, которая служит для задания масштаба 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. 4.12).

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. 4.13 в контекстном меню осциллограммы видна еще команда – **Autoscale** (**Автомасштабирование**). Эта же команда реализуется кнопкой **Autoscale** в панели инструментов окна осциллограммы. Данная команда устанавливает масштаб, при котором окно осциллограммы используется полностью. В подобном случае это означает, что осциллограмма будет иметь максимально возможный размер, как показано на рис. 4.13.

Итак, как и следовало ожидать, в результате моделирования получена синусоида с обрезанными на уровне 0.5 сверху и снизу вершинами. При этом результат

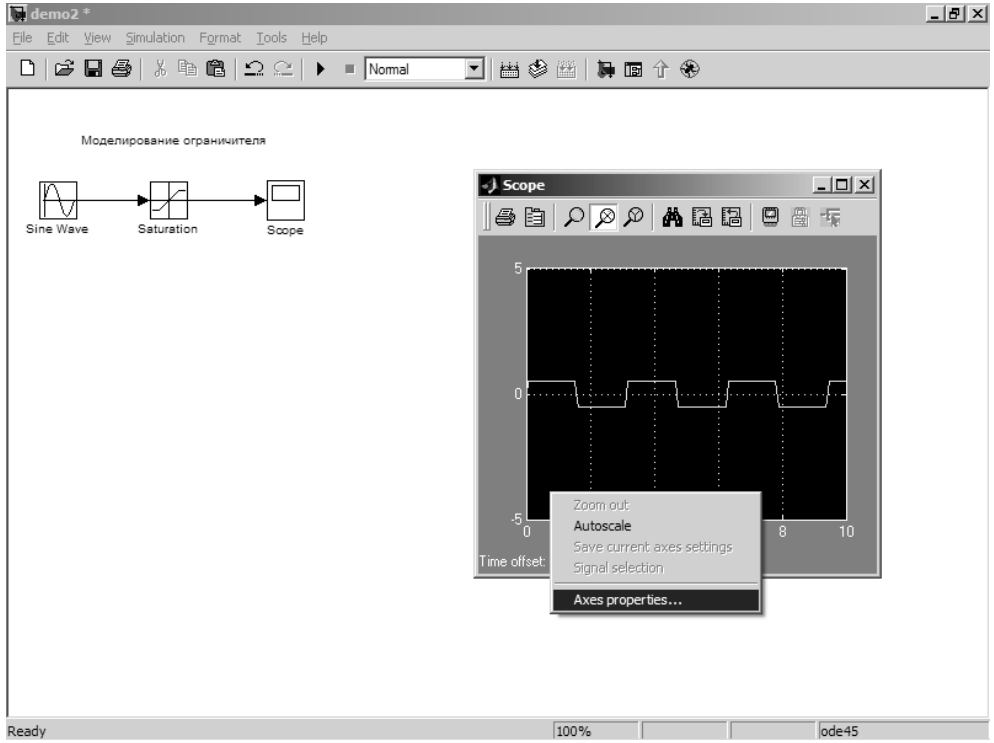


Рис. 4.11. Вывод контекстного меню окна осциллограммы

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.

4.2.4. 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 – рис. 4.14. Модель записывается в виде файла с расширением **.mdl**.

4.2.5. Модернизация и расширение модели

Теперь, после создания простой модели, можно попытаться модернизировать ее и дополнить, изменить параметры модели и т. д. На рис. 4.15, например, показана модель, в которой к источнику синусоидального сигнала подключены уже четыре нелинейных устройства. Это ограничитель пиков, нелинейность типа «мертвая

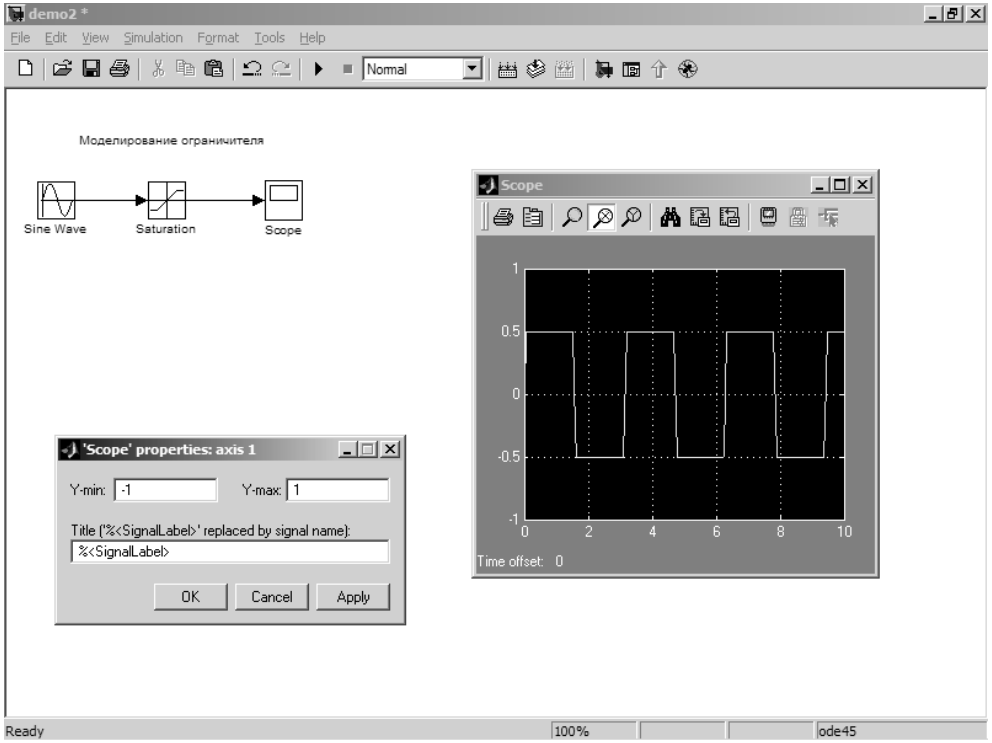


Рис. 4.12. Получение осциллограммы в новом масштабе

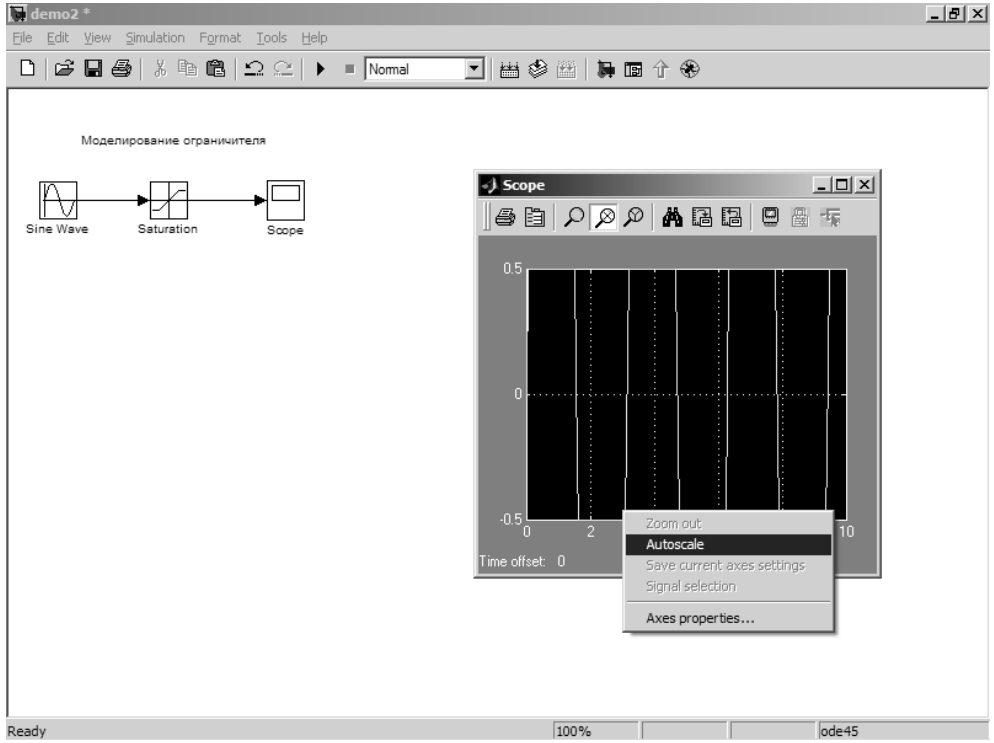


*Рис. 4.13. Осциллограмма после выполнения команды **Autoscale***

зона», квантующее устройство и релейный пороговый элемент (в электронике известный под именем триггера Шмитта). К каждому из них подключен свой осциллограф.

Как видно по рис. 4.15, 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.

4.3. Основные приемы подготовки и редактирования модели

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

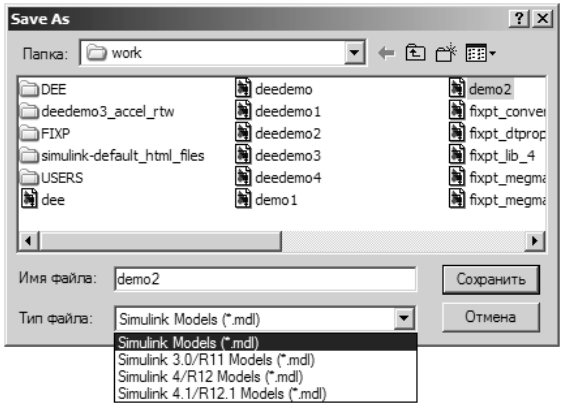


*Рис. 4.14. Запись модели командой **Save As***

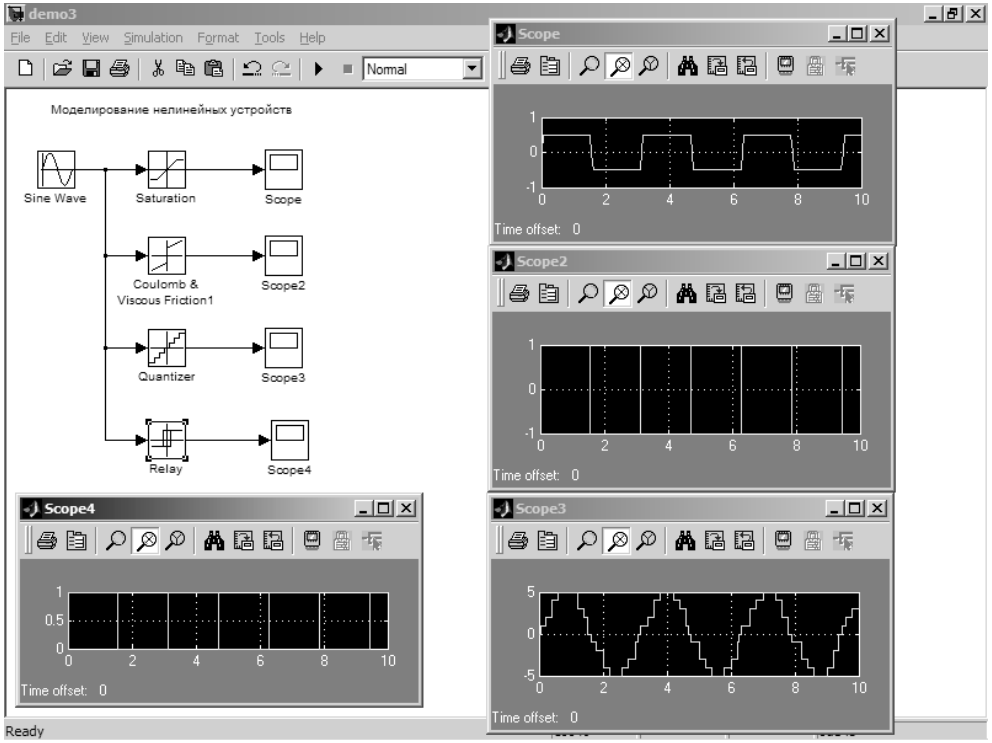


Рис. 4.15. Модель из трех нелинейных устройств

4.3.1. 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. 4.16).

Следует отметить, что для подготовки русскоязычных надписей нужно выбирать кириллические шрифты. Для этого нужно выделить текстовый блок и, установив в него курсор мыши, нажать правую клавишу мыши. Появится контекстное меню, представленное на рис. 4.17.

Рис. 4.16. Ввод текстового комментария

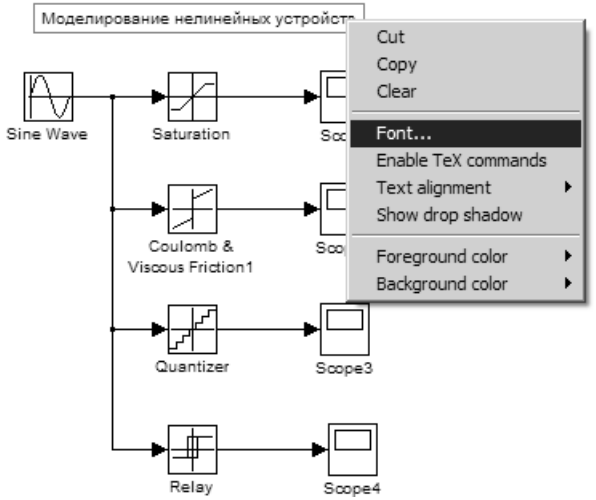


Рис. 4.17. Вывод контекстного меню текстового комментария

Исполнив команду контекстного меню **Font**, можно вывести окно выбора шрифта. Это окно показано на рис. 4.18.

Так же просто можно изменить подписи к блокам моделей. Для этого нужно установить мышь в область надписи и щелкнуть левой кнопкой мыши – в подпи-

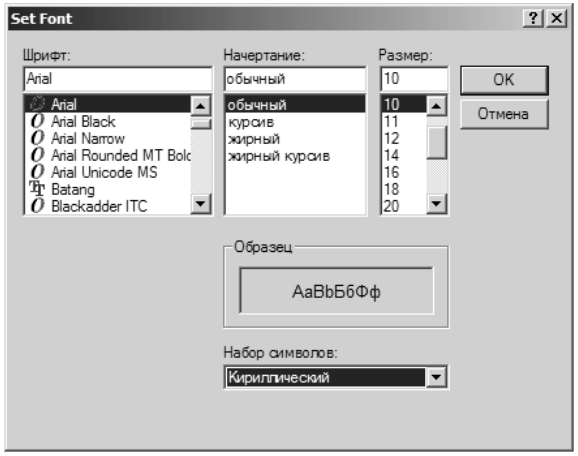


Рис. 4.18. Окно выбора шрифта текстового комментария

си появится курсор ввода, и ее можно будет редактировать. На рис. 4.19 показано изменение надписи в блоке осциллографа.

Рис. 4.19. Пример изменения подписи к блоку

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. Это справедливо даже при использовании MATLAB 7 + Simulink 6, несмотря на объявленную улучшенную поддержку интернациональных шрифтов (особенно в реализации с SP3).

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

4.3.2. 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**.

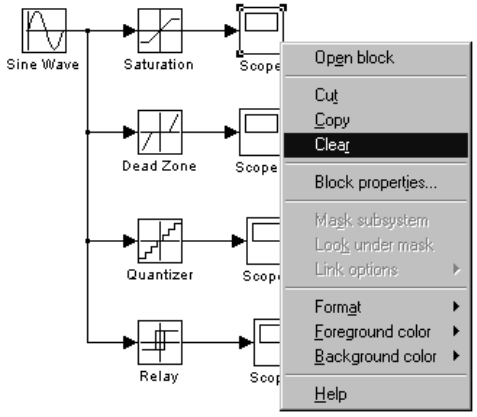


Рис. 4.20. Контекстное меню блока

Для стирания выделенного объекта можно вызвать команду **Clear** из меню **Edit** или из контекстного меню (рис. 4.20). Контекстное меню очень удобно тем, что для любого объекта оно выводит перечень команд, которые доступны в данном состоянии.

После исполнения команды **Clear** можно наблюдать удаление блока. Пример этого показан на рис. 4.21. Обратите внимание на то, что переименования оставшихся блоков при удалении одного из блоков не происходит. Следует учесть, что команда **Clear** стирает блок безвозвратно, то есть без помещения его в буфер обмена. Однако эту операцию можно отменить командой **Undo** в позиции **File** меню окна Simulink.

Для восстановления объекта в окне модели надо установить на нужное место его расположения курсор мыши и щелкнуть ее левой клавишей. После этого исполнение команды **Paste** из позиции **File** меню окна Simulink или из контекстного меню правой клавиши мыши помещает хранящийся в буфере объект (блок) в заданное место.

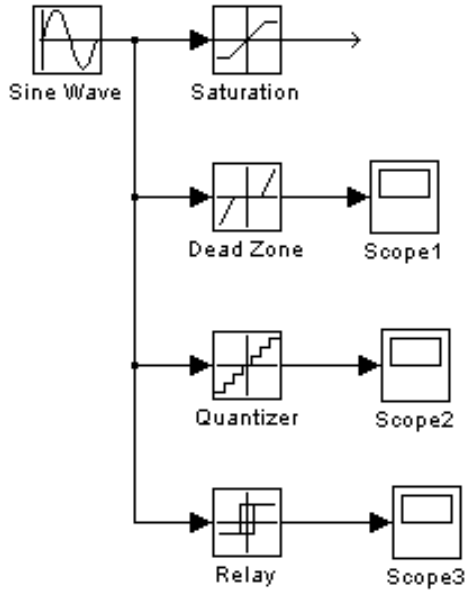


*Рис. 4.21. Модель с удаленным блоком **Scope***

4.3.3. 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 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 к входу следующего 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

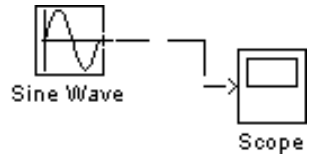


Рис. 4.22. Начало соединения блока источника с блоком осциллографа

нажатой левую кнопку мыши, надо плавно переместить курсор к входу следующего блока (рис. 4.22), где курсор мыши приобретет вид крестика из тонких сдвоенных линий.

Добившись протяжки линии к 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. 4.23).

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 4.24 поясняет этот процесс.

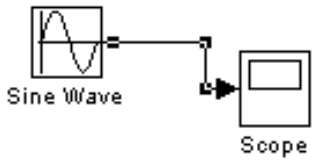


Рис. 4.23. Выделенное соединение

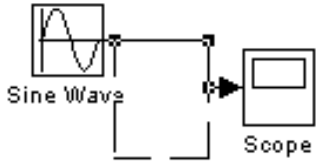


Рис. 4.24. Начало создания петли линии соединения

Создание петли линии заканчивается отпусканием левой кнопки мыши. Полученная таким образом линия показана на рис. 4.25.

Особо стоит отметить возможность задания наклонных линий соединений при нажатой клавише **Shift**. Пример такого соединения дан на рис. 4.26.

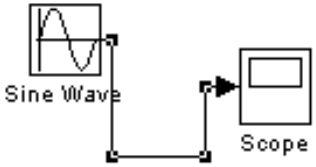


Рис. 4.25. Готовая петля соединения

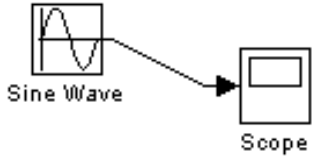


Рис. 4.26. Пример соединения с наклонной линией

4.3.4. Создание отвода линии

Часто возникает необходимость сделать отвод от уже созданной линии. Пример создания такого отвода иллюстрирует рис. 4.27. Заметим, что при нажатой клавише **Shift** отвод строится наклонными линиями.

В примере, показанном на рис. 4.28, использована модель интегратора, подключенного к выходу источника прямоугольных импульсов. Чтобы было можно наблюдать осциллограммы как на выходе источника, так и на выходе интегратора, в схему включен блок мультиплексора сигналов **Mux** с двумя входами. Чтобы

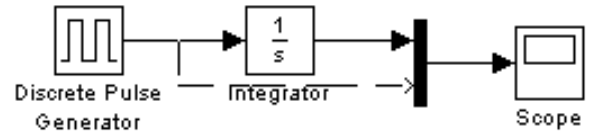


Рис. 4.27. Пример создания отвода (пунктирная линия)

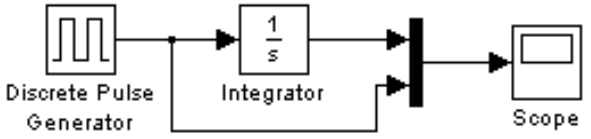


Рис. 4.28. Пример модели с отводом линии

подключить нижний вход к уже задействованному выходу источника, нам и понадобилось создать отвод линии.

Теперь можно запустить эту модель и посмотреть, какие сигналы действуют на выходах источника и интегратора. Результат запуска представлен на рис. 4.29.

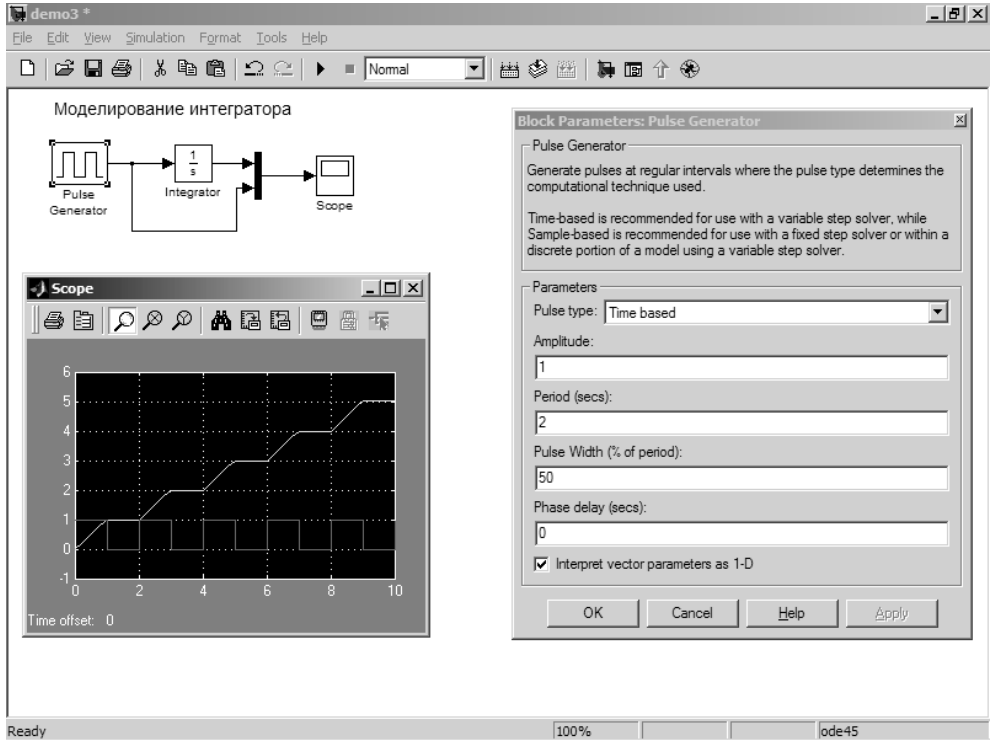


Рис. 4.29. Результат моделирования

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.

4.3.5. Удаление соединений

Для удаления соединительной линии достаточно выделить ее и выполнить команду **Clear** или **Cut**. Нужное соединение будет удалено. Напоминаем, что команда **Undo** позволяет восстановить удаленное соединение сразу после удаления.

4.3.6. 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. 4.30.

Увеличенный в размерах блок показан на рис. 4.31. Обратите внимание на то, что растягивается только графическое изображение (пиктограмма) блока, а размеры его названия в виде текстовой надписи не изменяются.

4.3.7. 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

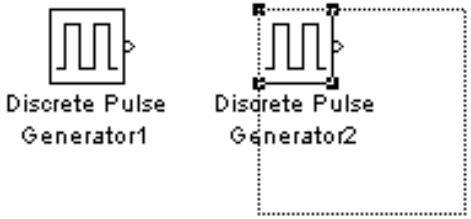


Рис. 4.30. Растяжение блока

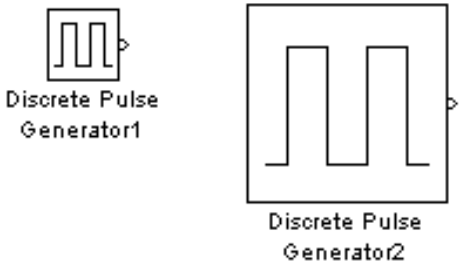


Рис. 4.31. Пример растяжения блока

новый блок, не разрушая его и не выполняя сложных манипуляций. Рисунок 4.32 показывает вставку блока дифференцирующего устройства между источником синусоидального сигнала и осциллографом.

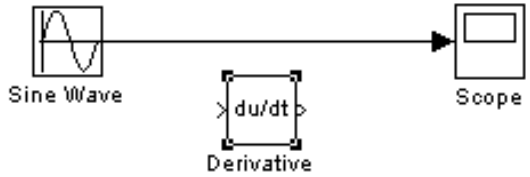


Рис. 4.32. Исходное соединение и блок для вставки

Результат вставки дифференцирующего устройства в соединение между источником и осциллографом показан на рис. 4.33.

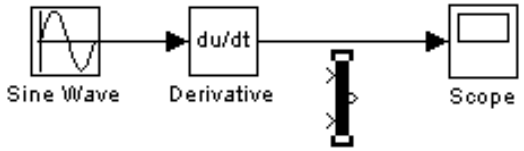


Рис. 4.33. Пример вставки блока в соединение

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.

4.3.8. 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. 4.34.

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 от $\sin(x)$ есть $\cos(x)$).

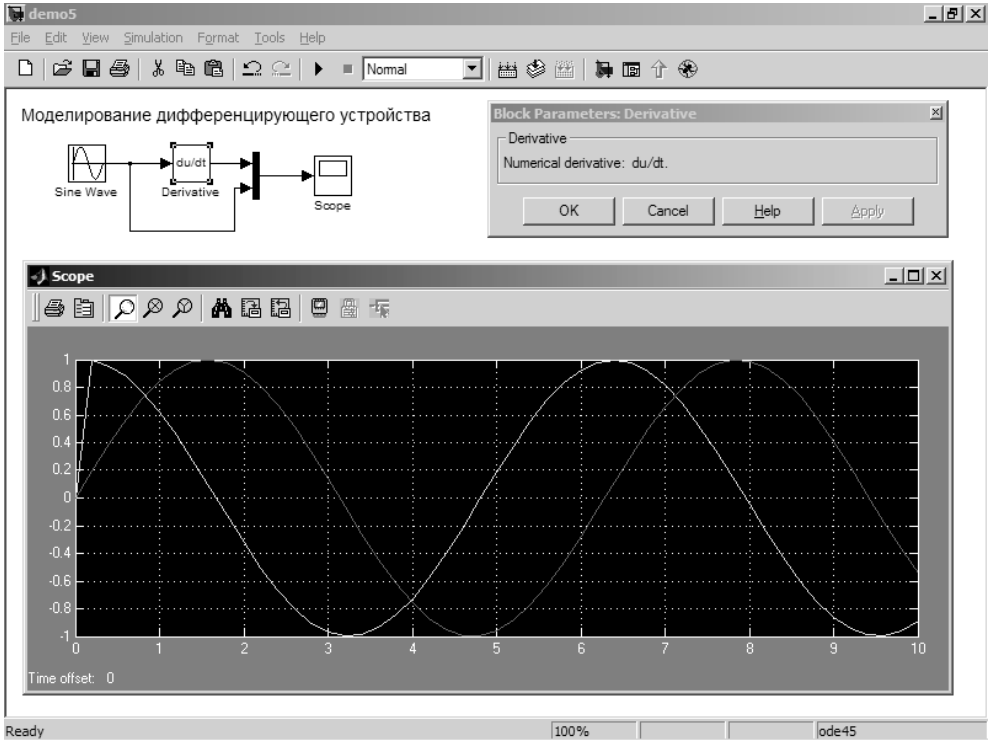


Рис. 4.34. Моделирование дифференцирующего устройства

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.

Обратите внимание, что в отличие от блока интегратора (рис. 4.29) блок цифрового дифференциатора не имеет настраиваемых параметров. Его окно параметров, показанное на рис. 4.26, является чисто информационным.

4.3.9. 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**.

4.4. Операции форматирования модели

4.4.1. 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.

4.4.2. Примеры форматирования модели

Команда **Format** ⇒ **Font** выводит окно с установками шрифта для текстовых надписей (рис. 4.18). Это позволяет форматировать текстовые блоки.

Рисунок 4.35 показывает операции, связанные с изменением положения блоков. Это команды **Flip block** и **Rotate block**.

Наконец, на рис. 4.36 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**.

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.

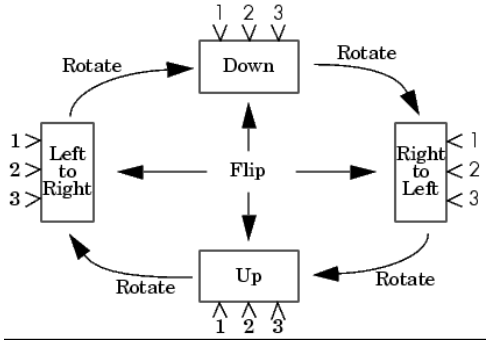


Рис. 4.35. Операции изменения положения блоков

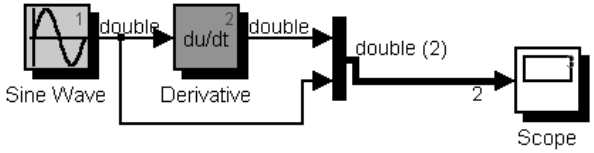


Рис. 4.36. Вид модели после операций форматирования

Урок 5

Блоки источников и получателей сигналов

Любая модель состоит как минимум из источника сигнала и его получателя. Simulink имеет обширные наборы источников и получателей сигналов общего назначения. С ними мы ознакомимся в этом уроке.

5.1. Источники простых сигналов и воздействий

5.1.1. Общий обзор источников

Окно раздела библиотеки **Sources** содержит пиктограммы источников воздействий произвольного характера. В электро- и радиотехнике их принято называть источниками электрических сигналов, но в физике, механике и в других областях науки и техники такое название не очень подходит. В них природа воздействий может быть самой разнообразной, например в виде перепада давления или температуры, механического перемещения, интенсивности звуковой волны и т. д. Учитывая это, мы будем называть соответствующие компоненты просто источниками.

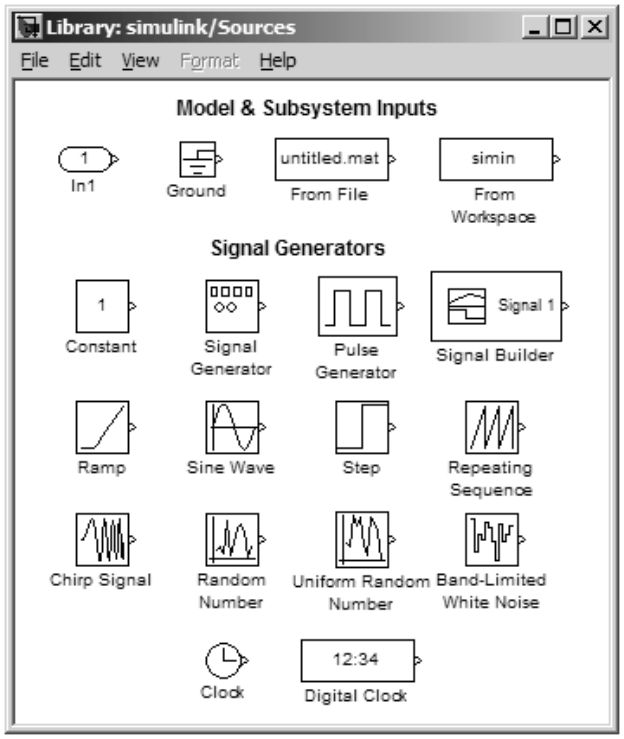


Рис. 5.1. Окно с перечнем источников сигналов и воздействий

Окно раздела библиотеки с источниками **Sources** представлено на рис. 5.1. Для его получения нужно выделить мышью позицию в браузере библиотек и открыть окно, используя команду **Open** в контекстном меню правой клавиши мыши.

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.

Пиктограммы блоков сигналов имеют вид прямоугольников с изображением временной зависимости сигналов на выходе сигналов. Это облегчает восприятие модели.

5.1.2. Источник постоянного воздействия Constant

Источник постоянного воздействия **Constant** задает константу или вектор констант. 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 [-1 0 1 2 3.14] задает вектор из пяти констант со значениями –1, 0, 1, 2 и 3,14. Этот случай представлен на рис. 5.3.

Установка флажка **Interpret vector parameters as 1-D** имеется и у других источников, и в дальнейшем мы ее специально отмечать не будем.

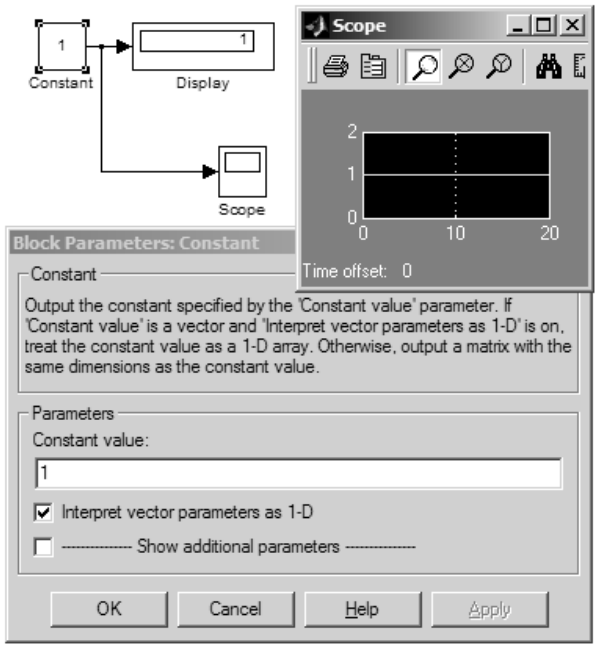


Рис. 5.2. Источник постоянного воздействия с одним значением

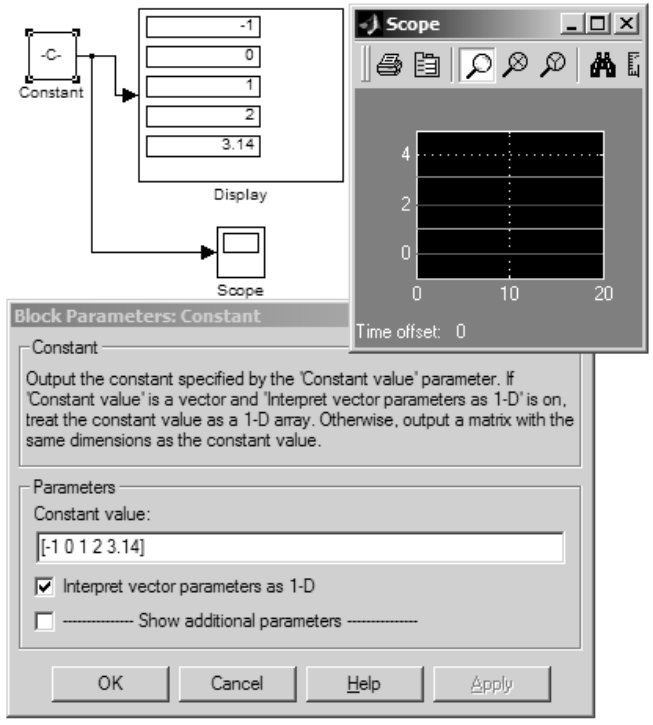


Рис. 5.3. Источник постоянного воздействия со многими значениями

Источник постоянного воздействия может выдавать такие воздействия с разным форматом их представления. Возможные форматы представлены в открытом списке на рис. 5.4. Выбран формат double – данные удвоенной длины в формате с плавающей точкой. Цифровой индикатор может выводить данные также с различным форматом. Это тоже показано на рис. 5.4.

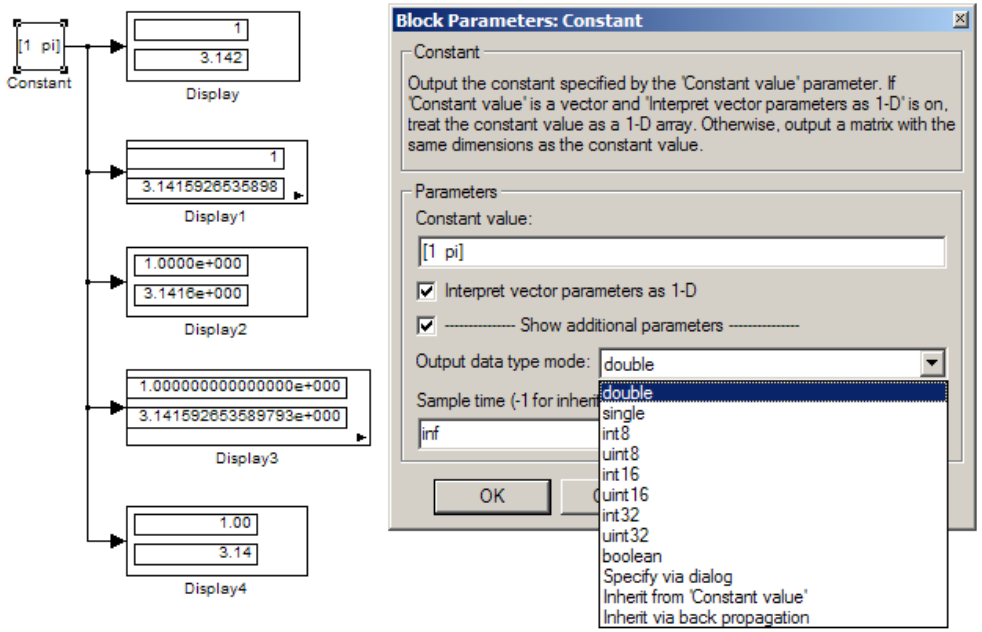


Рис. 5.4. Иллюстрация задания и вывода значений двухэлементного вектора с различными форматами

5.1.3. Источник синусоидального воздействия Sine Wave

Синусоидальное воздействие – одно из самых распространенных. Мы уже описывали источник синусоидального сигнала в уроке 14 – см. рис. 14.10, на котором показано и окно параметров этого источника. Он характеризуется амплитудой **Amplitude**, смещением по вертикали **Bias**, 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).

Создание синусоидального сигнала возможно двумя алгоритмами, которые выбираются списком **Sine types**:

- `Time based` – по текущему времени для непрерывных сигналов или по конечно-разностному алгоритму для дискретных сигналов;

- `Sample based` – по такту дискретности и числу расчетных шагов за один период сигнала.

При формировании синусоидального сигнала по текущему времени используется формула:

$f(t) = A\sin(2\pi ft + \varphi) + b,$

где A – амплитуда сигнала, f – частота, t – текущее время, φ – фаза и b – сдвиг сигнала (постоянная составляющая). В условиях ограниченной точности вычислений такой расчет при больших временах может приводить к заметным ошибкам.

При конечно-разностном алгоритме используется следующее представление сигнала:

$$\begin{bmatrix} \sin(t+\Delta t) \\ \cos(t+\Delta t \end{bmatrix} = \begin{bmatrix} \cos(\Delta t)\cdot\sin(\Delta t) \\ -\sin(\Delta t)\cdot\cos(\Delta t) \end{bmatrix} \cdot \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}.$$

Этот алгоритм обеспечивает меньшую погрешность округления для больших времен.

При формировании по количеству тактов за период используется следующее соотношение:

$f(t) = A\sin(2\pi kfT + \varphi) + b = A\sin((2\pi k + l)/N) + b,$

где T – такт дискретности, N – количество тактов в секунду, l – начальная фаза, выраженная в тактах, $k = 0, 1..., N-1$ – номер текущего шага.

5.1.4. Источник нарастающего воздействия Ramp

Источник линейно нарастающего воздействия вида $F(t) = k*t + i$ и окно его параметров представлены на рис. 5.5. 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 i.

Как правило, указываются значения параметров, принятые по умолчанию. Обратите внимание на то, что за пределами отрезка времени моделирования [0, 10] воздействие не определено.

5.1.5. Источник одиночного перепада Step

Источник воздействия в виде одиночного перепада показан на рис. 5.6.

Параметры источника:

- **Step time** – время появления перепада (скачка);
- **Initial value** – начальное значение воздействия (до перепада);
- **Final value** – конечное значение воздействия (после перепада);

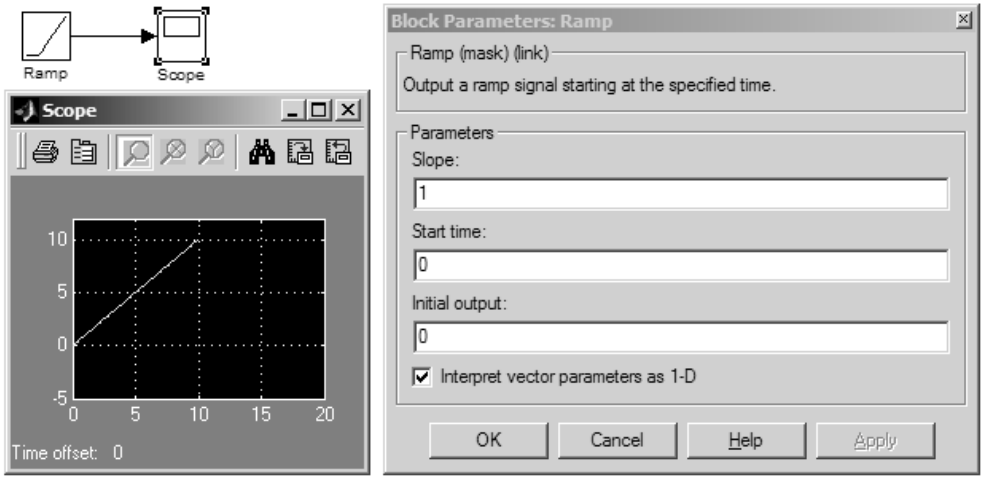


Рис. 5.5. Источник линейно нарастающего воздействия

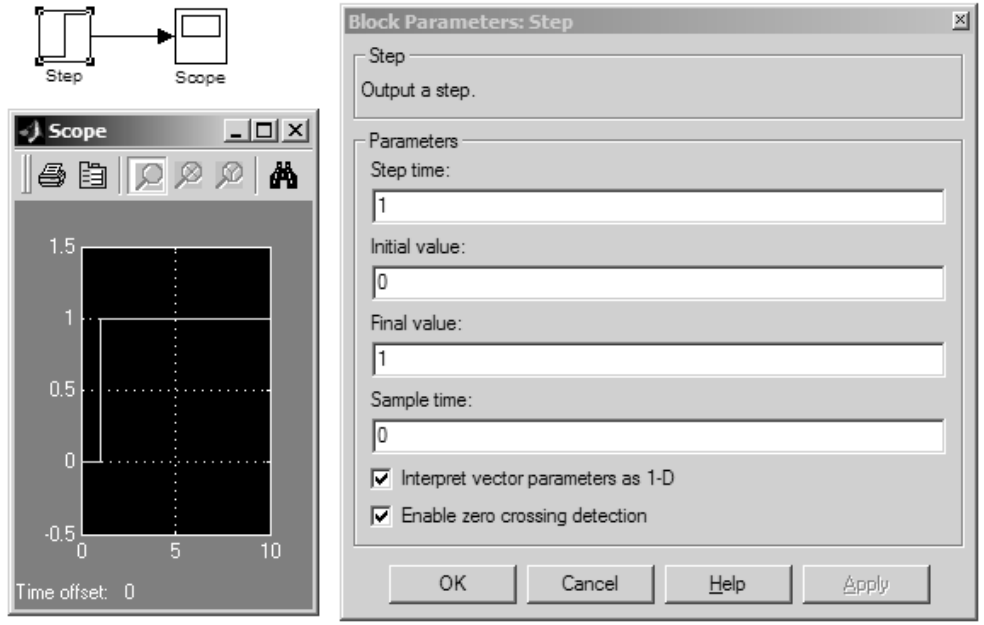


Рис. 5.6. Источник одиночного перепада

- **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.

5.1.6. Источник прямоугольных импульсов Pulse Generator

Источник дискретных импульсов **Pulse Generator** служит для создания прямоугольных однополярных импульсов (см. рис. 5.7).

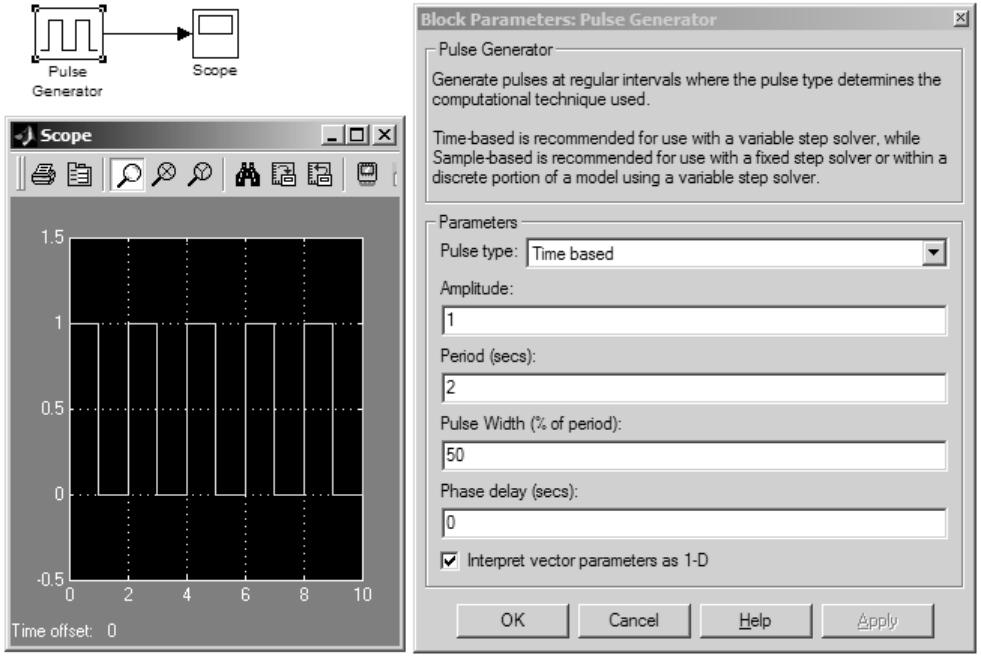


Рис. 5.7. Источник прямоугольных импульсов

Параметры источника по умолчанию обеспечивают формирование однополярных прямоугольных импульсов – см. осциллограмму на рис. 5.7. Можно устанавливать следующие параметры сигнала этого источника:

- **Amplitude** – амплитуда;
- **Period** – период (кратный эталонному времени);
- **Pulse width** – ширина импульсов (в процентах от периода);
- **Phase delay** – фазовая задержка (кратная эталонному времени);
- **Simple time** – эталонное время.

5.1.7. Земля Ground

Блок заземления **Ground** служит для заземления входов блоков или иных точек модели – рис. 5.8. Его можно рассматривать как источник нулевого воздействия.

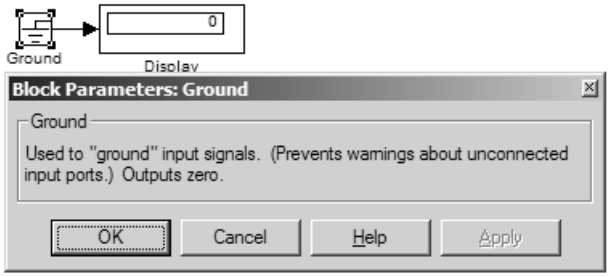


Рис. 5.8. Источник нулевого воздействия – земля

5.2. Источники шумовых воздействий

5.2.1. 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. 5.9.

Специфическими параметрами этого источника являются среднее значение сигнала **Mean** и среднеквадратическое отклонение **Variance**. Назначение двух других параметров (см. окно установки параметров на рис. 5.9) уже отмечалось.

5.2.2. 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 **Minimum**. Применение этого источника и окно установки его параметров представлены на рис. 5.10.

Осциллограмма сигнала также представлена на рис. 5.10. Надо помнить, что она не будет повторяться в ваших экспериментах, поскольку сигнал источника является случайным. Параметр **Initial seed** задает начальное значение сигнала.

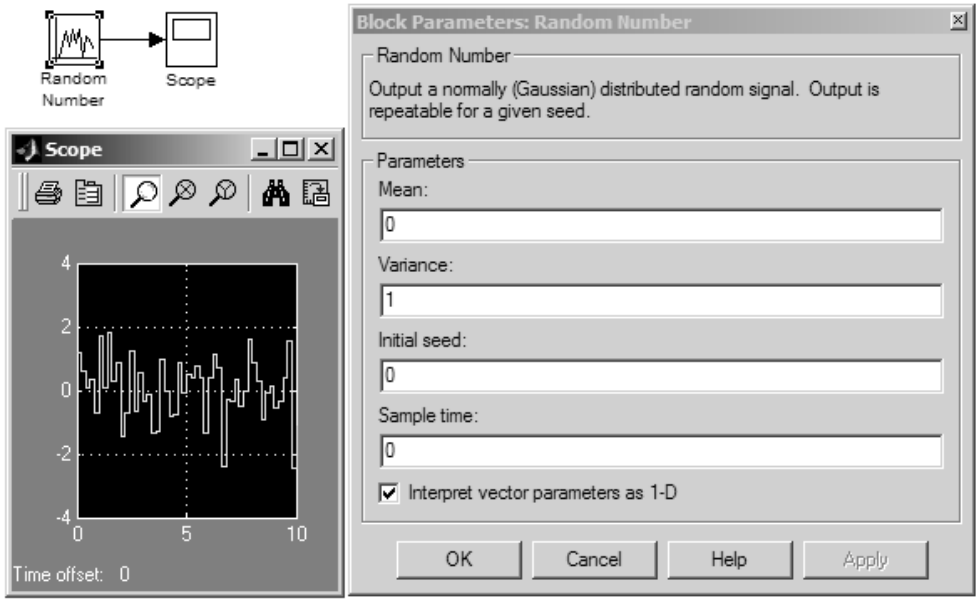


Рис. 5.9. Источник случайного сигнала с нормальным распределением

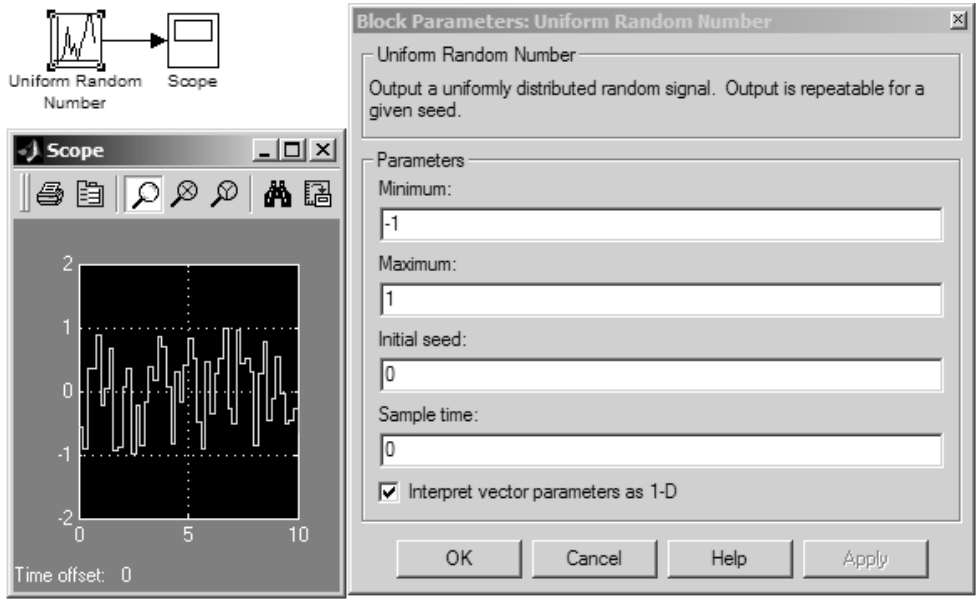


Рис. 5.10. Источник случайного сигнала с равномерным распределением

5.2.3. Генератор белого шума Band Limited White Noice

Генератор белого шума **Band Limited White Noise** служит для создания шумового сигнала с заданной мощностью, равномерно распределенной по частоте. Рисунок 5.11 показывает работу этого генератора и окно установки его параметров.

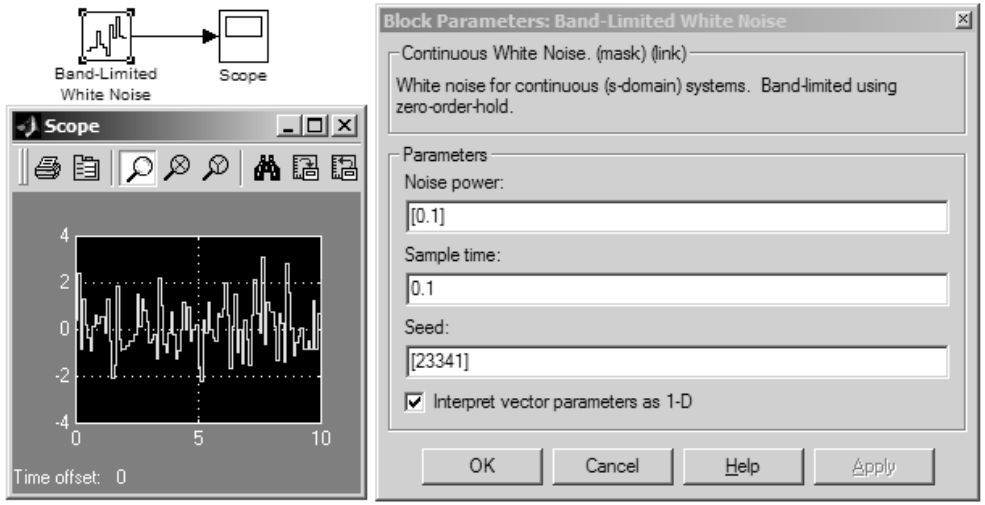


Рис. 5.11. Генератор белого шума

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.

5.3. Источники сложных сигналов

5.3.1. Повторяющаяся последовательность Repeating Sequence

Источник повторяющихся последовательностей **Repeating Sequence** служит для генерации последовательности, заданной векторами моментов времени и значений сигнала в эти моменты времени – рис. 5.12. Узловые точки сигнала, заданные этими векторами, соединяются отрезками прямых, то есть между ними задается линейная интерполяция.

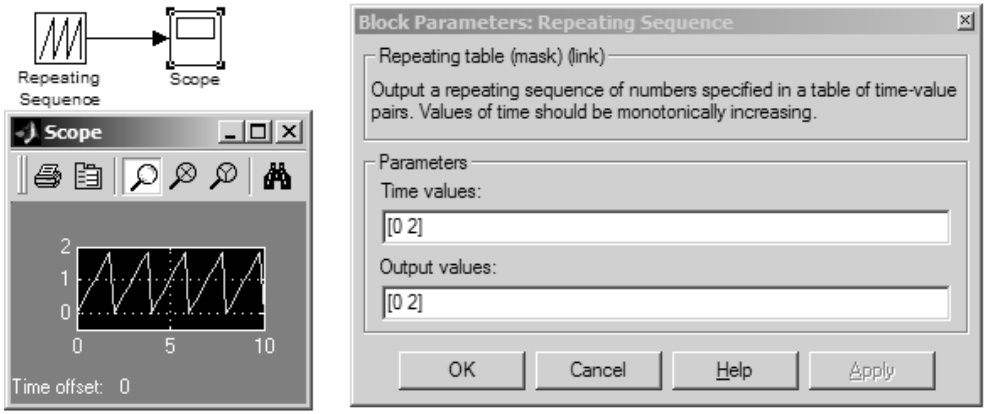


Рис. 5.12. Источник повторяющейся последовательности

Параметры источника этого типа:

- **Time values** – вектор значений времени;
- **Output values** – вектор выходных значений.

Форма заданного сигнала отображается внутри пиктограммы. Следует отметить, что этот источник обеспечивает весьма большие возможности в генерации импульсных сигналов различного вида. По умолчанию формируется пилообразный сигнал – рис. 5.12.

5.3.2. Сигнал-генератор Signal Generator

Источник воздействия типа **Signal Generator** служит для создания одного из четырех наиболее распространенных типов сигналов:

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

Рисунок 5.13 показывает действие этого источника при моделировании пилообразного сигнала и окно его параметров. Выбор формы сигнала осуществляется в списке **Wave form**.

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 в этом разделе.

5.3.3. Генератор нарастающей частоты Chirp Generator

Генератор нарастающей частоты **Chirp Generator** создает почти синусоидальные колебания, частота которых увеличивается до некоторого момента времени (см. рис. 5.14).

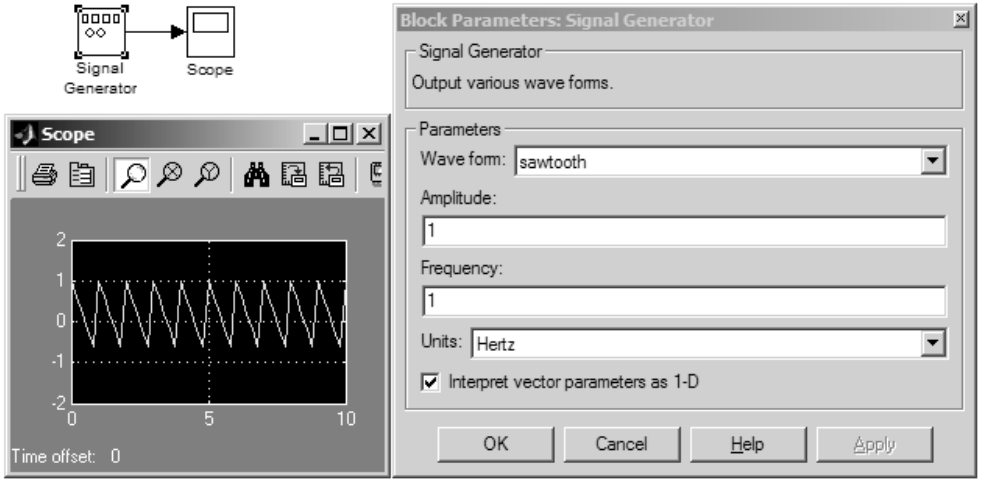


*Рис. 5.13. Источник типа **Signal Generator***

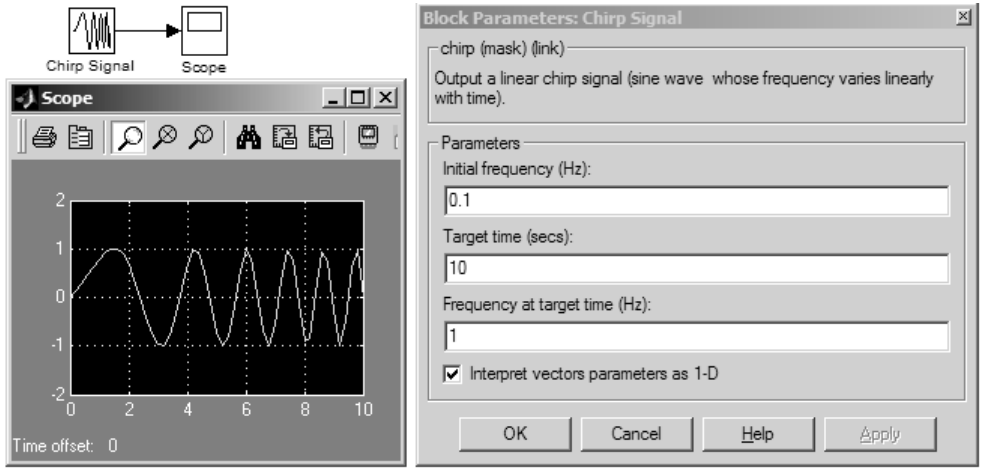


Рис. 5.14. Генератор нарастающей частоты

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. 5.14 заданы параметры, отличные от принятых по умолчанию.

5.3.4. Конструктор сигналов

Конструктор сигналов – это, конечно, полезное дополнение в Simulink 5. Он представлен блоком **Signal Builder**, имеющим единственный выходной порт. Сигнал на выходе этого порта имеет вид импульса, форму которого можно менять с помощью окна, представленного на рис. 5.15.

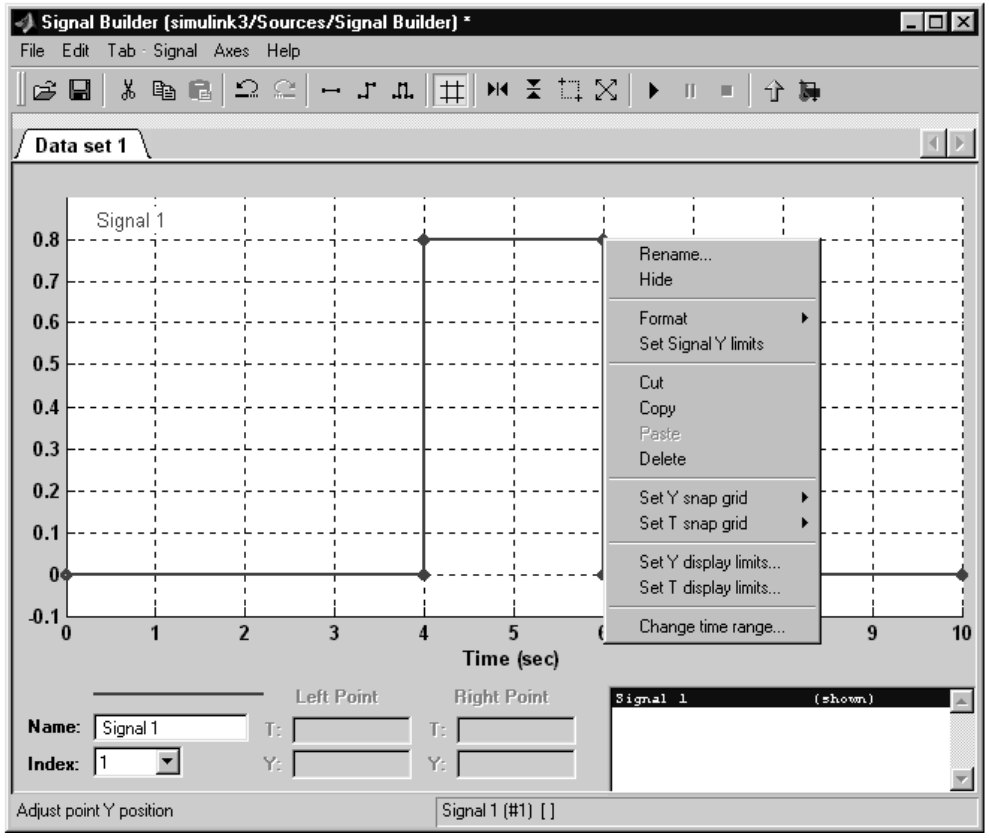


Рис. 5.15. Окно конструктора сигналов Simulink 5

Блок **Signal Builder** можно перетащить мышью в окно модели и использовать этот блок как источник сигнала с устанавливаемой пользователем формой сигнала. По умолчанию конструктор создает сигнал в виде прямоугольного импульса, который виден в окне конструктора, – рис. 5.16. Однако, используя мышь, можно перемещать линии импульса и придавать им наклон – то есть менять форму исходного импульса. После этого можно наблюдать форму сигнала, подключив к выходу блока блок осциллографа. Все эти возможности иллюстрирует рис. 5.16.

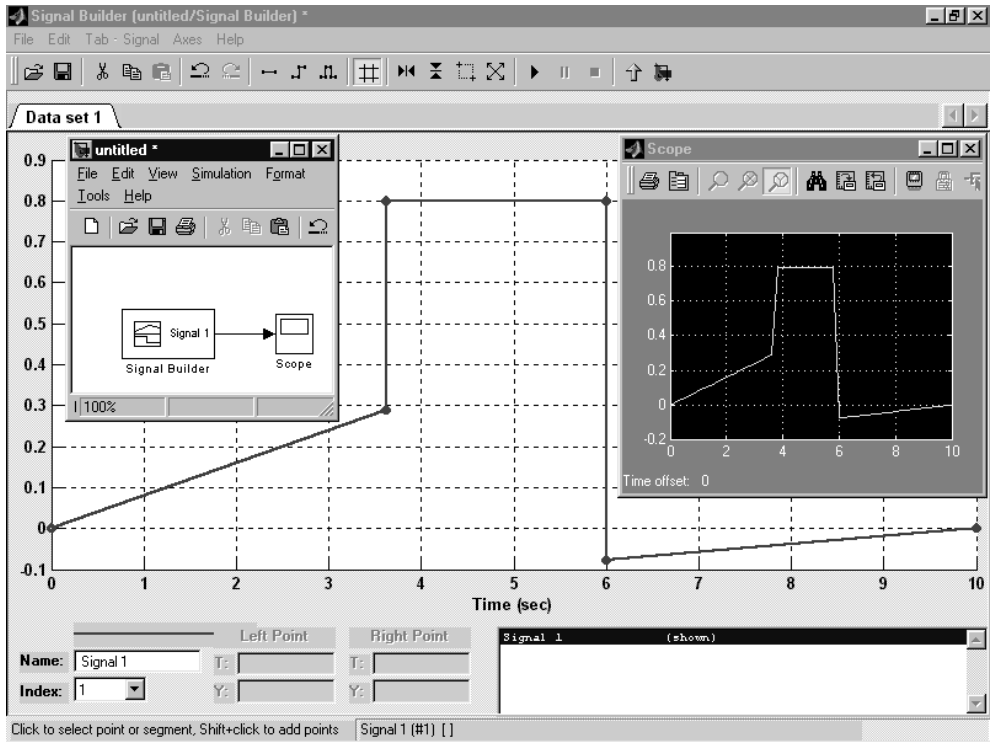


Рис. 5.16. Применение конструктора сигналов

5.4. Источники системных данных

5.4.1. 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** (см. рис. 5.17).

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

5.4.2. 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. 5.18 показана работа источника **Digital Clock**.

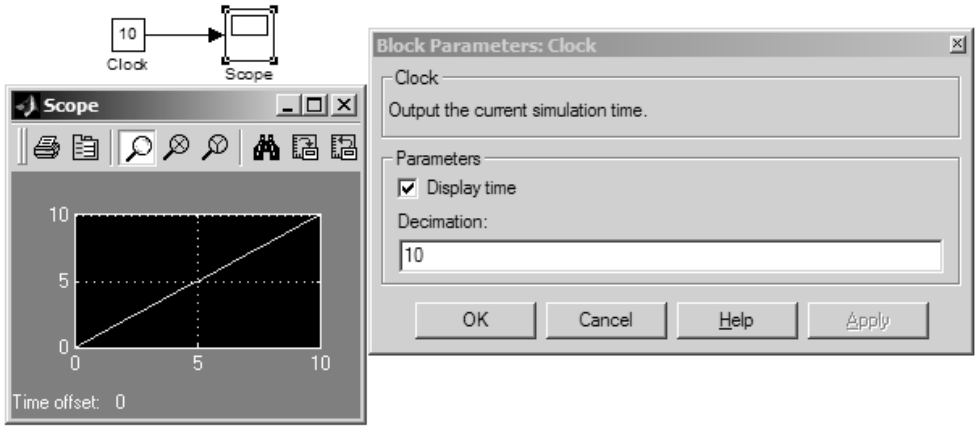


Рис. 5.17. Источник времени моделирования

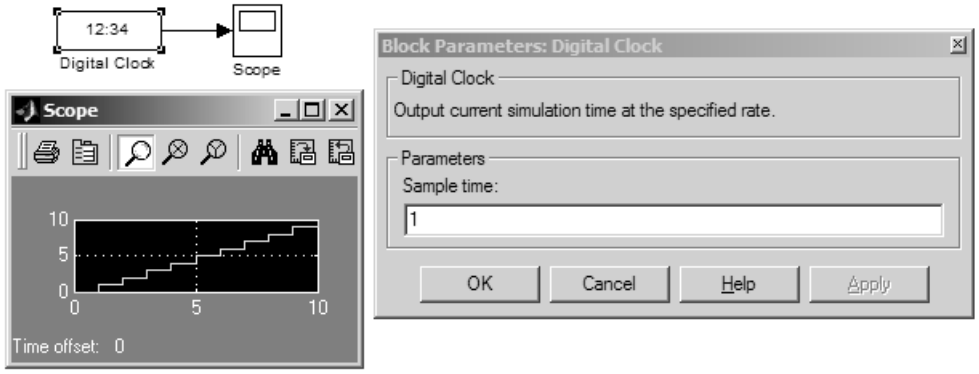


Рис. 5.18. Цифровой источник времени моделирования

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

5.4.3. Блок получения данных из файлов From File

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

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

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. Пример применения этого блока дан на рис. 5.19.

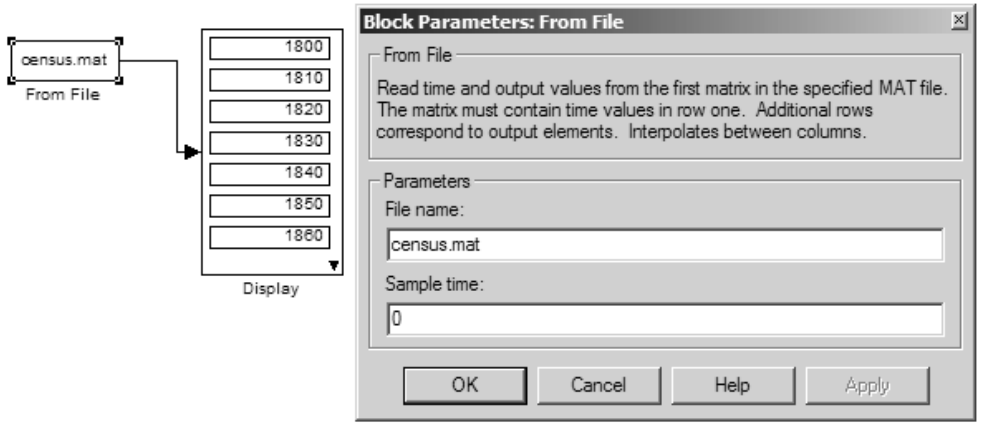


Рис. 5.19. Источник сигнала из файла

5.4.4. Блок получения данных из рабочего пространства From Workspace

Блок **From Workspace** служит для получения данных из рабочего пространства. В качестве параметров задаются формат матрицы данных (по умолчанию [T, U]) и эталонное время **Sample time** (по умолчанию 0). Рисунок 5.20 поясняет применение этого блока.

Кроме того, имеется список для задания конечного значения путем:

- **Interpolate data** – экстраполяции данных;
- **Setting to zero** – установки на нуль;
- **Hold final data value** – задержки последнего значения данных, что бывает нужно для его представления регистрирующими блоками;
- **Cyclic Repetition** – циклического повторения.

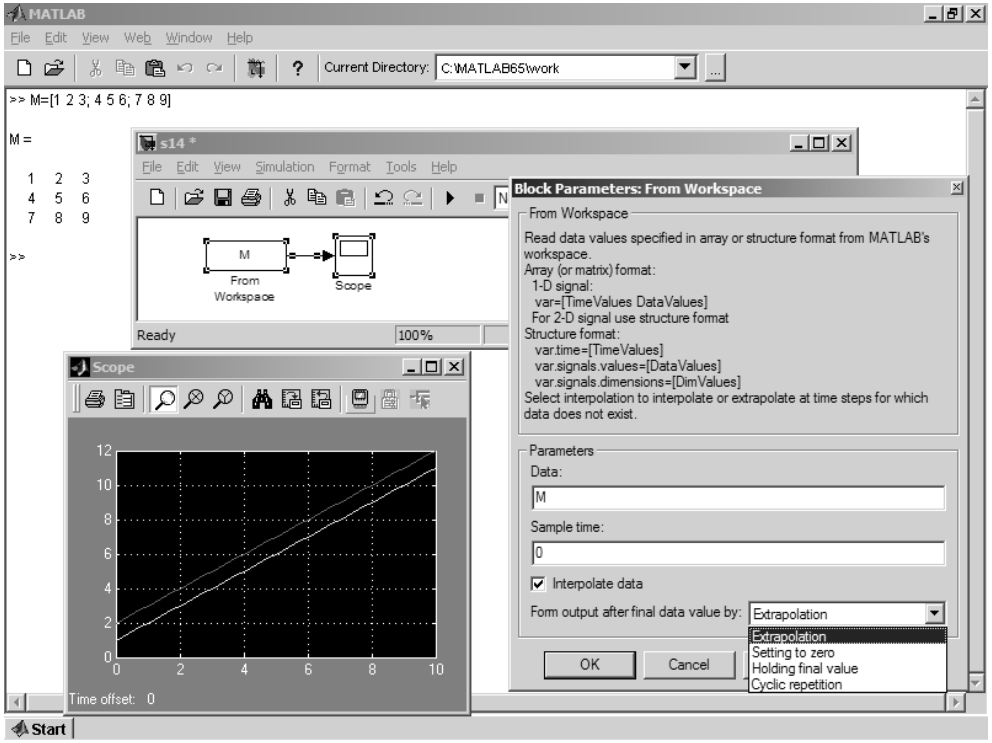


*Рис. 5.20. Пример применения блока **From Workspace***

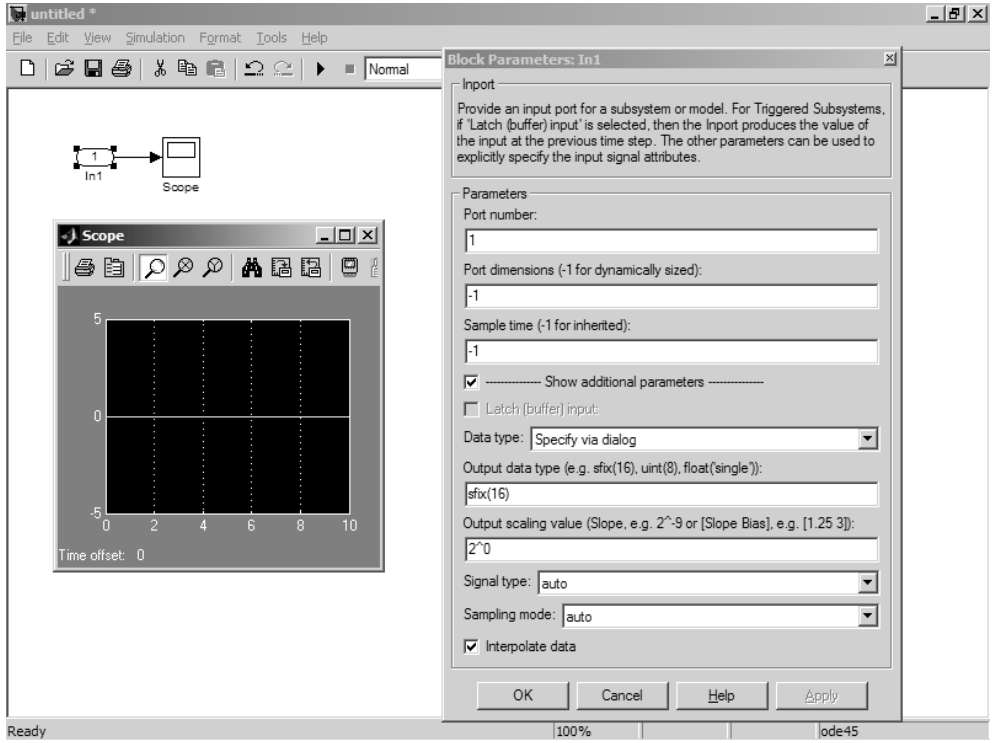


*Рис. 5.21. Блок входа **In***

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 далее.

5.4.5. Блок входа In

Блок входа **In***, где * означает номер блока, служит для организации входов подсистем. Для них эти блоки являются входами. Блоки входа имеют сквозную нумерацию, причем при удалении блоков она автоматически возобновляется. Простой пример применения блока **In** представлен на рис. 5.21. Осциллограф показывает отсутствие сигнала на выходе блока **In1**, а окно параметров блока позволяет устанавливать его параметры.

Основными параметрами блока `In` являются номер порта `Port Number`, размерность порта `Port dimension` (размерность определяется автоматически, если этот параметр равен –1) и **Simple time**. Установка опции **Show additional parameters** расширяет окно параметров блока `In` до показанных на рис. 5.21 размеров и позволяет задавать ряд дополнительных параметров:

- `Data types` – выбор из списка типов данных (при выборе **Specify via dialog** появляются зоны дополнительных параметров, указанных ниже);
- `Output data types` – задание нужного типа данных (например, utit(8), sfix(16) и т. д.);
- `Output scaling mode` – режим масштабирования выходного сигнала (`Best precision: Vector-vise` – наивысшая точность, `Use specified scaling` – использование заданного масштаба);
- `Output scale value` – величина масштаба (примеры задания см. на рис. 5.21). Эта опция возможна при использовании режима масштабирования `Use specified scaling`;
- `Signal type` – выбор типа сигнала (auto, real или complex);
- `Sampling mode` – выбор режима работы (auto, Sample based или Frame based);

- `Interpolate data` – интерполяция данных.

К более детальному описанию блока **In** мы вернемся при рассмотрении применения подсистем (субблоков).

5.5. Виртуальные регистраторы

5.5.1. Обзор блоков приема данных

Библиотека **Sinks** содержит блоки получателей данных. На рис. 5.22 показано окно раздела библиотеки пакета Simulink с блоками получателей данных.

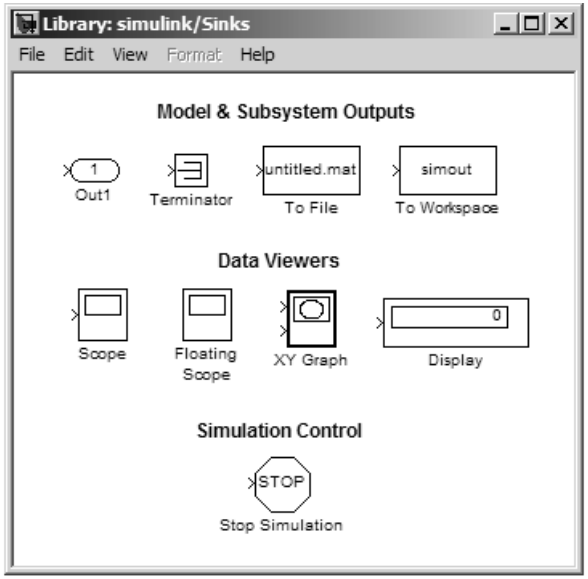


Рис. 5.22. Окно блоков приема данных

Библиотека блоков приема данных содержит три группы блоков:

- `Data Viewers` – регистрирующие блоки для просмотра данных;
- `Model & Subsystems Outputs` – блоки организации выходов в моделях и подсистемах;
- `Simulation Control` – блок организации контроля моделирования **Stop**.

Наличие регистрирующих компонентов – важный фактор качественной визуализации результатов моделирования. 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;

- **Floating Scope** – «плавающий» осциллограф;
- **XY Graph** – графопостроитель в системе полярных координат;
- **Display** – устройство вывода на экран дисплея.

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.

- **Out*** – выход с номером * для организации интерфейса с подсистемой;
- **Terminator** – нагрузка/заглушка для неиспользуемых выходов;
- **To File** – устройство записи данных в файл;
- **To Workspace** – устройство записи в переменную рабочего пространства.

Перейдем к рассмотрению виртуальных регистраторов.

5.5.2. 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. Приведем еще один пример (рис. 5.23) на контроль осциллографом треугольного сигнала.

Осциллограф имеет свою панель инструментов, показанную на рис. 5.24 и содержащую следующие кнопки:

1. **Print** – печать содержимого окна осциллографа;
2. **Parameters** – открытие окна параметров осциллографа;
3. **Zoom** – увеличение масштаба по осям X и Y одновременно;
4. **Zoom X-axis** – увеличение масштаба по горизонтальной оси X;
5. **Zoom Y-axis** – увеличение масштаба по вертикальной оси Y;
6. **Autoscale** – автоматическое масштабирование, позволяющее наблюдать осциллограмму с максимальным размером;
7. **Save current axes setting** – сохранение текущих установок параметров окна;
8. **Restore saved current axes setting** – установка ранее сохраненных настроек;
9. **Floating scope** – превращение осциллографа в «плавающий»;
10. **Lock/Unlock axes selection** – фиксация/разрыв связи между текущей координатной системой окна и отображаемым сигналом (только в случае «плавающего» осциллографа);
11. **Signal selection** – выбор сигналов для отображения (только в случае «плавающего» осциллографа).

На рис. 5.23 окно параметров осциллографа показано с открытой вкладкой **General**, содержащей основные параметры:

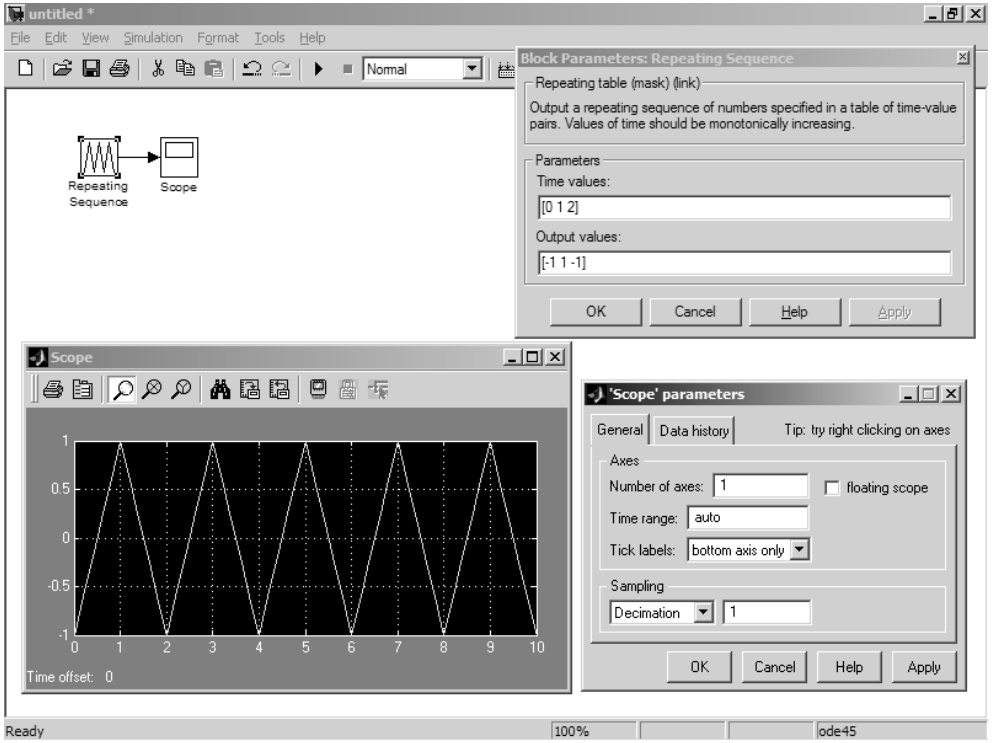


Рис. 5.23. Работа с одноканальным виртуальным осциллографом

Рис. 5.24. Панель инструментов виртуального осциллографа

- **Number of axes** – число осей (каналов) осциллографа;
- **Time range** – пределы временного интервала;
- **Tick labels** – вывод/скрытие отметок по осям;
- **Sampling** – установка временных соотношений (Decimation – кратность вывода данных, по умолчанию 1, или Simple Time – в тактах эталонного времени, по умолчанию 0).

Параметр **Number of axes** позволяет превратить одноканальный осциллограф в многоканальный путем указания нужного числа входов. При этом осциллограф приобретает несколько входных портов, к которым можно подключать различные сигналы. Пример применения осциллографа в таком режиме представлен на рис. 5.25.

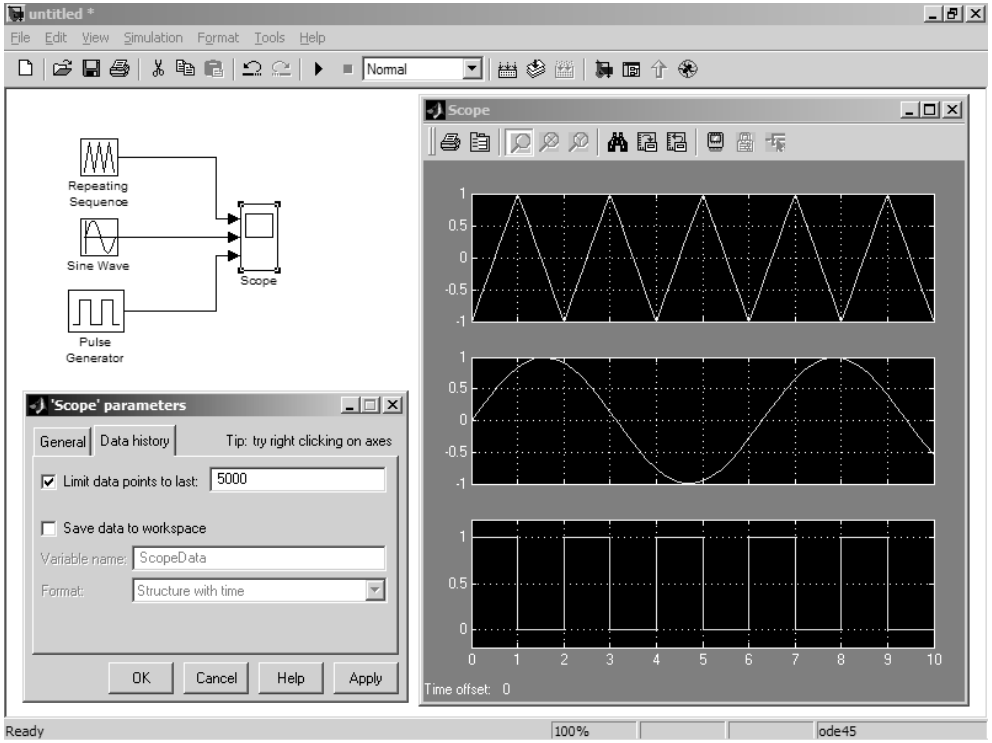


Рис. 5.25. Пример работы осциллографа в трехканальном варианте

На рис. 5.25 показано также окно параметров осциллографа с открытой вкладкой **Data history**. Здесь можно задать максимальное число точек осциллограмм для хранения и параметры хранения осциллограмм в рабочем пространстве системы MATLAB.

При использовании виртуального осциллографа 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.

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.

5.5.3. Виртуальный «плавающий» осциллограф

Включение опции **floating scope** превращает осциллограф в «плавающий». При этом входы отключаются, а связи обозначаются пунктирными линиями. «Плавающий» осциллограф есть и как отдельный блок в составе блоков регистрирующих устройств. Он отличается от «обычного» тем, что у него отсутствуют в явном виде входы. Это упрощает построение моделей с большим числом осциллографов за счет отсутствия соединений между выходами тех или иных сигналов и входами осциллографа.

Рисунок 5.26 показывает применение «плавающего» осциллографа. Здесь моделируется ограничитель, содержащий источник синусоидального сигнала и блок его ограничения, нагруженный на терминатор, – эквивалент нагрузки. Под ними выведен «плавающий» осциллограф без входов. Пустив модель, можно настроить осциллограф на показ нужных сигналов.

Для настройки осциллографа прежде всего надо вывести его окно, наведя на блок осциллографа курсор мыши и дважды щелкнув левой клавишей мыши. Затем нужно выделить соответствующее окно осциллографа и выделить сигнал, ко-

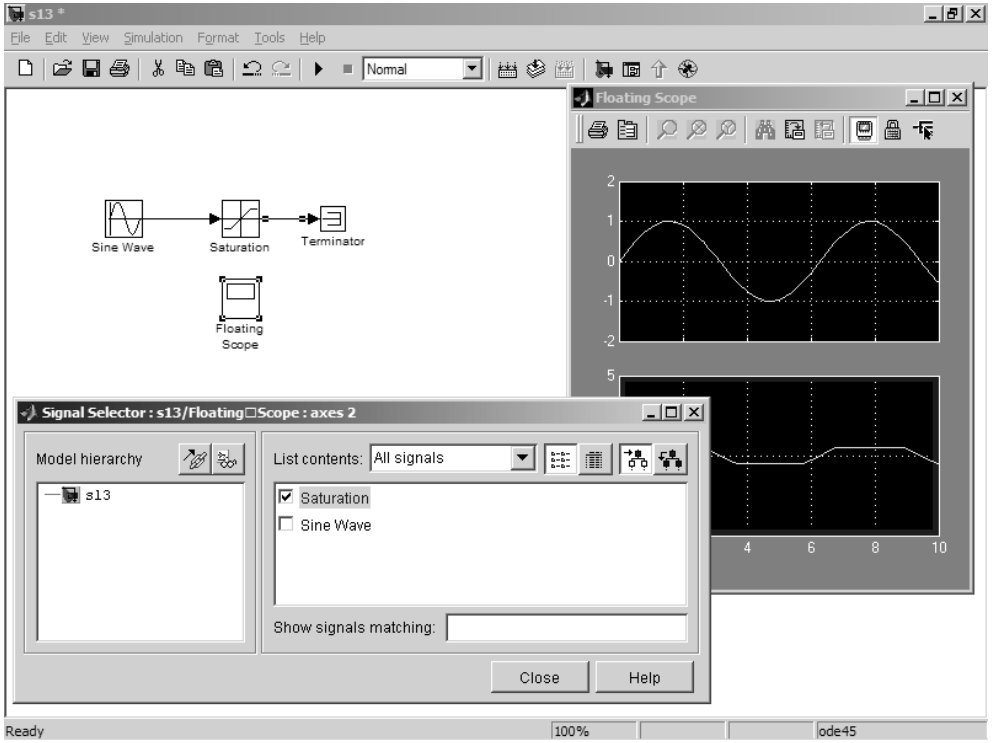


Рис. 5.26. Пример работы с «плавающим» осциллографом

торый должен быть в нем представлен. Для выделения окна необходимо подвести к нему курсор мыши и щелкнуть один раз левой клавишей мыши. Выделенное окно обрамляется рамкой из жирной синей линии. На рис. 5.26 это показано для нижнего окна «плавающего» осциллографа.

Для выделения сигнала надо нажать кнопку `Signal Selection` (она последняя в панели инструментов осциллографа – рис. 5.24). Можно также исполнить команду с тем же названием из контекстного меню правой клавиши мыши. Появится окно селекции сигнала, показанное на рис. 5.25 снизу. Левая часть окна дает схему (иерархию) выходов, а правая – список выходов на текущей ветви схемы (в данном случае она одна, а выходов – два). Указав нужный выход, надо закрыть окно, активизируя кнопку `Close`.

5.5.4. 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. 5.27.

На рис. 5.27 представлен пример построения фигур Лиссажу – они образуются при подаче на вход графопостроителя двух синусоидальных сигналов с частотами 2 Гц и 3 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.

Графопостроитель можно использовать и для построения функций вида $y(x)$ или $f(t)$. Пример такого построения дан на рис. 5.28. Здесь строится график синусоиды, причем на вход X графопостроителя подается линейно-нарастающий сигнал времени. Он осуществляет временную развертку.

Обратите внимание на установку параметра Simple Time равным –1. Это позволяет получать синусоиду с достаточно большим числом точек. При этом ее график получается плавным.

5.5.5. Дисплей Display

Виртуальный дисплей – устройство представления цифровой информации. Это одно из самых простых устройств. На рис. 5.2, 5.3 и 5.4 уже приводились примеры применения дисплея для отображения значений констант и векторов. Чтобы все компоненты вектора были видны, дисплей в окне Simulink растянут мышью по горизонтали (его можно растягивать и по вертикали).

Окно параметров дисплея показано на рис. 5.29. В нем устанавливается формат отображения данных **Format**, прореживание входных данных **Decimation** и в тактах эталонного времени **Sample time**. Можно также выбрать представление вещественных данных в формате с плавающей точкой (флажок **Floating point**).

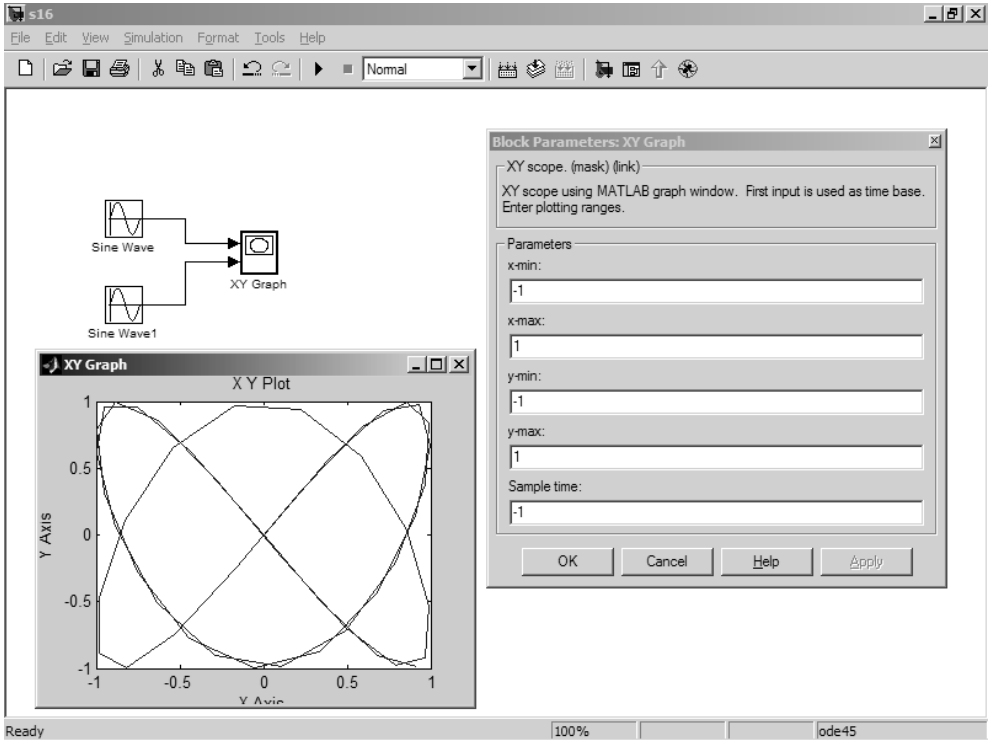


Рис. 5.27. Применение виртуального графопостроителя для построения фигуры Лиссажу

Дисплей обеспечивает динамическое отображение данных, то есть можно наблюдать их изменение в процессе моделирования.

На рис. 5.29 представлен вывод числа 100*pi в разных форматах. Заметим, что для этого приходится расширять блок дисплея. При использовании нескольких дисплеев они нумеруются как Display, Display1, Display2 и т. д. Опция-флажок делает дисплей «плавающим», то есть отключает вход дисплея (соединение при этом показывается пунктирной красной линией). Параметр Decimation при значении 1 выводит все данные вектора, при значении 2 – каждый второй, при 3 – каждый третий элемент вектора и т. д.

5.6. Другие блоки группы Skins

5.6.1. Заглушка Terminator

Если в модели встречаются отключенные выводы блоков, то может наступить ошибка моделирования. Для предотвращения таких ситуаций служит специальный блок – заглушка, или терминатор `Terminator`. Этот блок имеет только вход

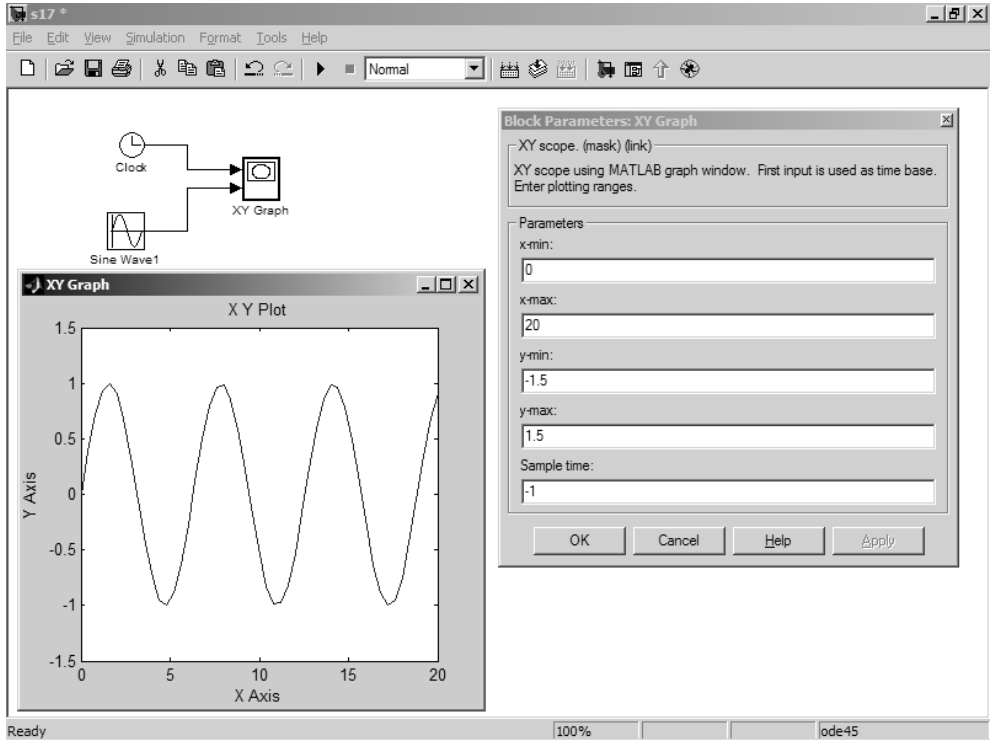


Рис. 5.28. Применение виртуального графопостроителя для построения графика синусоиды

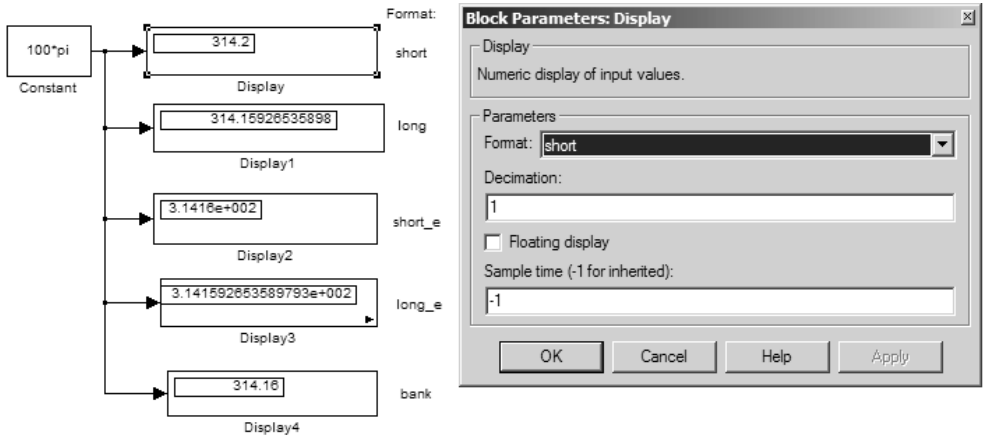


Рис. 5.29. Вывод чисел дисплеем в разных форматах

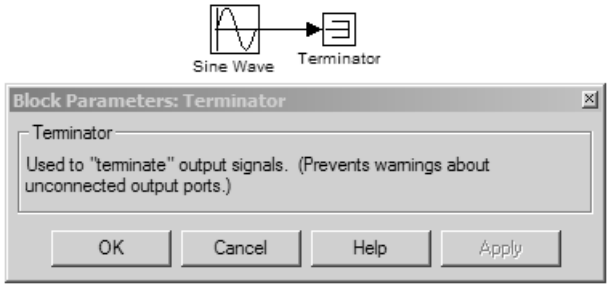


Рис. 5.30. Пример применения блока терминатора

и чисто информационное окно параметров – рис. 5.30. Другой пример применения этого блока дан на рис. 5.26.

5.6.2. Задание выхода Out* и создание подсистемы

Блок **Out***, где * означает номер блока, служит для задания порта выхода в подсистемах. Напомним, что блок **In*** служит для создания порта входа. Рассмотрим модель модуляции синусоидального сигнала с частотой 10 Гц и амплитудой 1 В синусоидальным сигналом с частотой 1 Гц и амплитудой 0,5 В, наложенным на постоянный уровень 1. Модель, построенная с применением математических блоков умножения и сложения, представлена на рис. 5.31.

Рисунок 5.31 показывает результат моделирования и окно параметров моделирования, в котором задан режим моделирования с фиксированным шагом 0,01 и применением для моделирования метода Рунге-Кутта, – решатель ode4. Приведена осциллограмма полученного амплитудно-модулированного сигнала с коэффициентом модуляции, равным 0,5.

Теперь выделим часть модели, которую можно оформить в виде подсистемы. Выделение показано на рис. 5.32 и осуществляется мышью при нажатой левой клавише. Нужный блок выделяется прямоугольником с пунктирными черными линиями.

Для создания подсистемы достаточно исполнить команду **Create Subsystem** (создать подсистему) с позиции **Edit** меню. Выделенная часть окажется преобразованной в блок подсистемы с именем **Subsistem**, имеющий два входа и один выход – рис. 5.33. Двойной щелчок левой клавиши мыши, при курсоре, установленном на блоке, выводит окно подмодели. Оно, как и окно параметров блока **Out1**, и осциллограмма работы этой системы с подсистемой показаны на рис. 5.33.

В окне параметров подсистемы устанавливаются номер порта и значение сигнала перед началом моделирования. Как нетрудно заметить, в данном случае блоки **In1**, **In2** и **Out1** подсистемы были созданы автоматически. Разумеется, можно модернизировать или создавать заново подсистему и редактировать ее точно так же, как и основную модель. При записи модели с подсистемой она записывается вместе с последней.

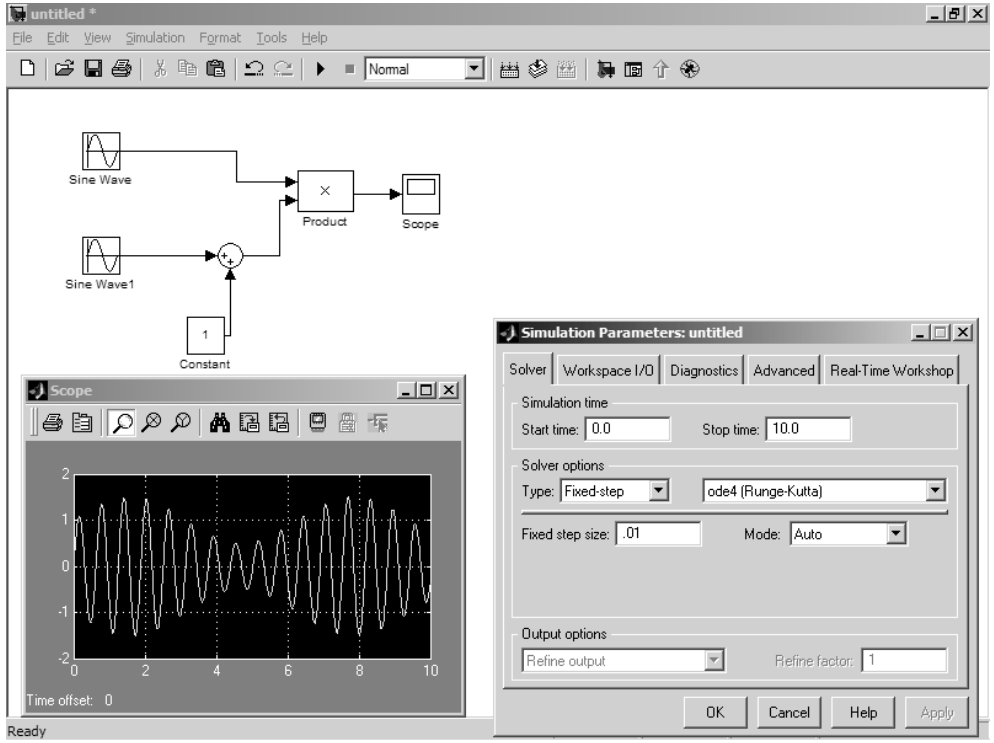


Рис. 5.31. Модель осуществления амплитудной модуляции

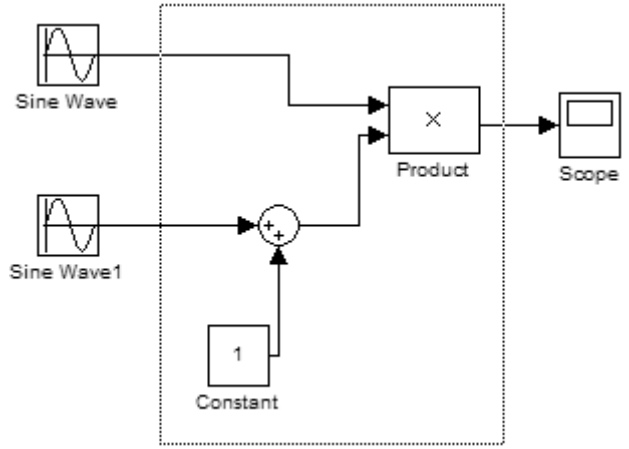


Рис. 5.32. Выделение будущей подсистемы

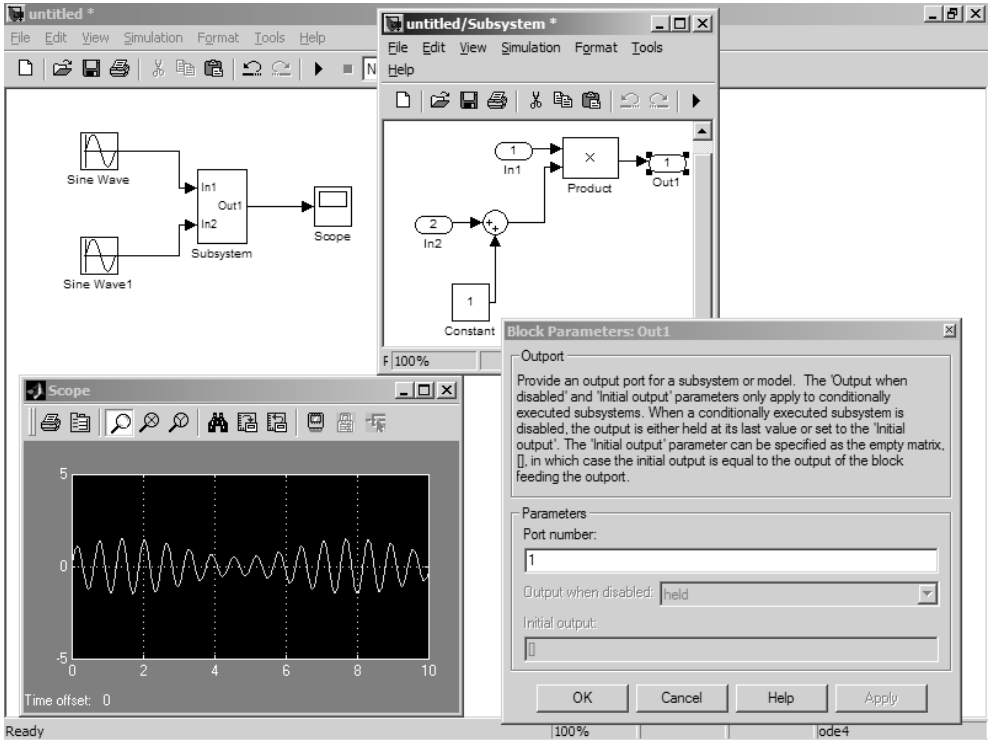


Рис. 5.33. Модель с подсистемой

5.6.3. 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. 5.34. На вход блока подан сигнал со значением 0. Поэтому моделирование идет непрерывно, и осциллограф показывает линейно нарастающее значение текущего времени.

Теперь измените константу 0 на 1. Блок **Stop** при этом запретит моделирование. Поэтому в данном случае осциллограф уже не будет фиксировать нарастание времени – моделирование прерывается в самом начале. 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. 5.34) чисто информационное.

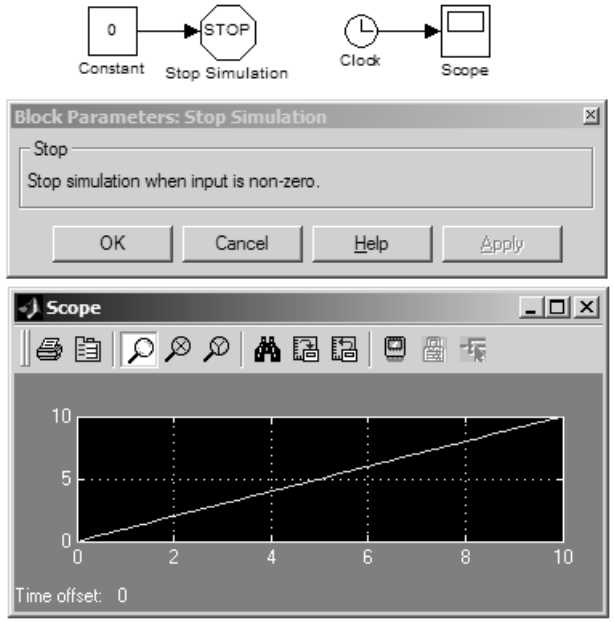


Рис. 5.34. Применение блока остановки моделирования

5.6.4. Блок сохранения данных в файле To File

Блоки сохранения данных в файле **To File** записывают входные данные в виде матриц соответствующего формата – (5.1), такого же, как для блока **From File**. 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**). Пример применения и окно параметров блока показаны на рис. 5.35. В данном случае в файл mysin.mat записываются отсчеты синусоидального сигнала. Файл (если не задан иной путь) располагается в папке WORK системы MATLAB.

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

Рисунок 5.36 показывает применение блока **From File** для считывания созданного ранее файла mysin.mat. После считывания с помощью осциллографа строится график синусоиды.

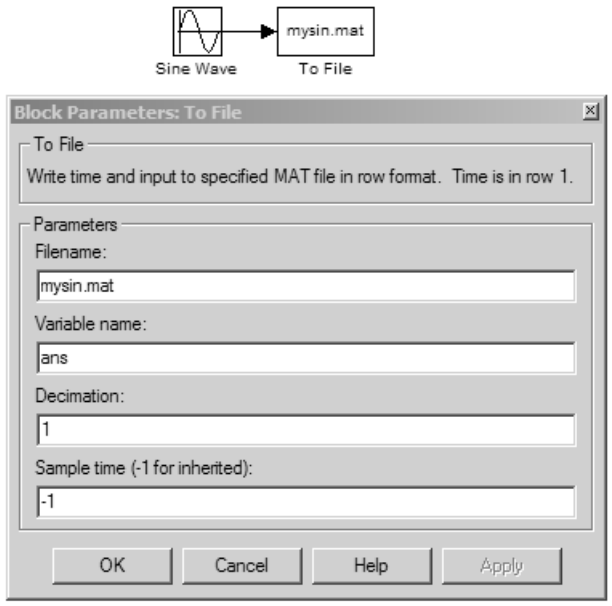


*Рис. 5.35. Пример применения блока **To File***

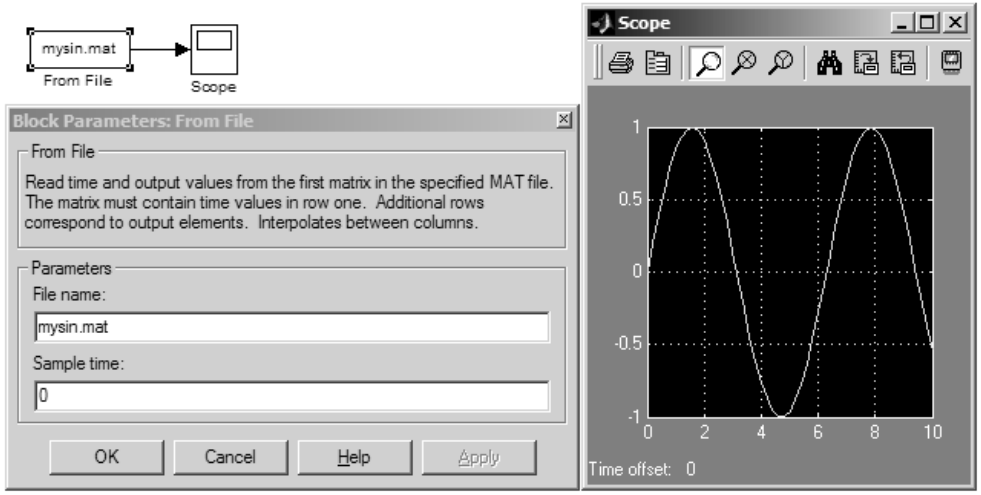


Рис. 5.36. Пример считывания файла mysin.mat

5.6.5. Блок сохранения данных в рабочем пространстве To Workspace

Блок **To Workspace** записывает указанную матрицу (но без строки отсчетов времени) в рабочее пространство. Пример применения блока представлен на рис. 5.37. В данном случае в рабочее пространство записывается вектор rs пилообразного сигнала, создаваемого блоком **Repeat Sequence**. Обратите внимание на появление в окне браузера рабочего пространства (оно видно слева) вектора rs после пуска примера.

На рис. 5.37 показано также 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 считывания данных рабочего пространства служит рассмотренный выше блок **From Workspace**.

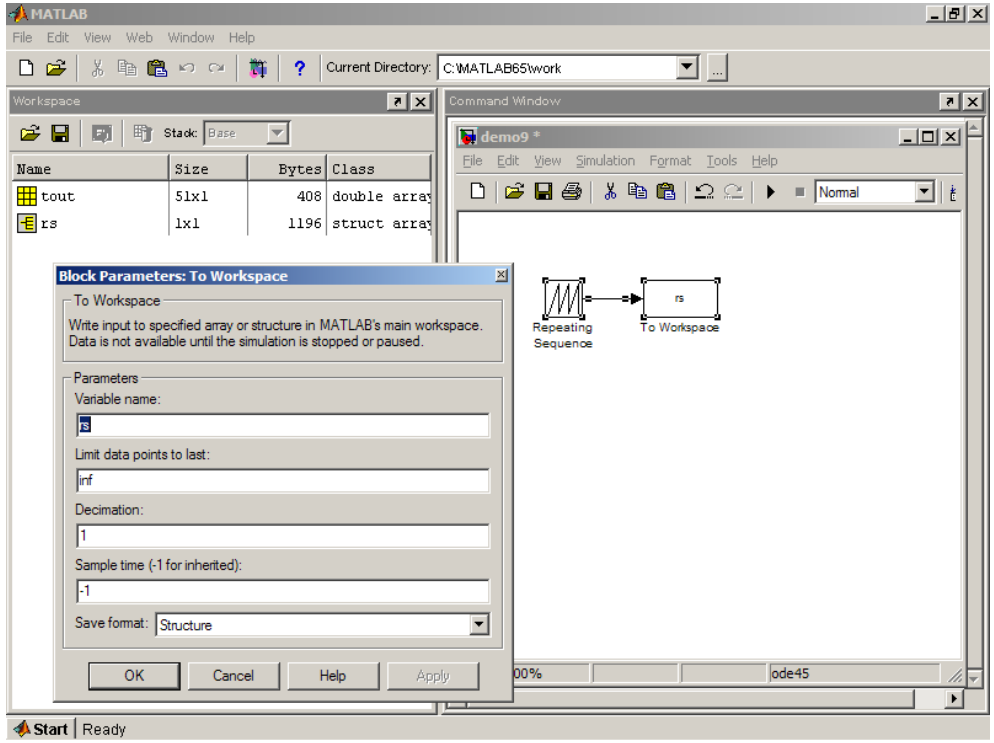


*Рис. 5.37. Пример применения блока **To Workspace***

5.7. Библиотека Signal Routing

5.7.1. Обзор библиотеки Signal Routing

Окно библиотеки **Signal Routing** представлено на рис. 5.38. Ранее большинство блоков этой библиотеки входило в библиотеку **Signal & Systems** расширения Simulink 4. В Simulink 5 последняя уже отсутствует.

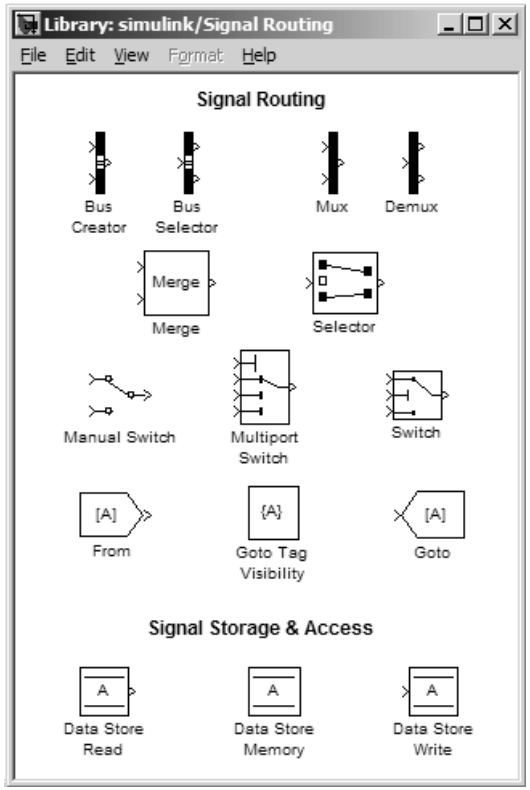


Рис. 5.38. Окно с блоками библиотеки

Библиотека содержит два раздела: **Signal Routing** и **Signal Storage & Access**. В раздел **Signal Routing** входят следующие блоки:

- **Bus Creator** – создание шины;
- **Bus Selector** – селекция сигналов шины;
- **Mux** – мультиплексирование ряда входов;
- **Dmux** – демультиплексирование сигналов шины;
- **Manual Switch** – ручной переключатель сигналов;
- **Multiport Switch** – многопортовый переключатель сигналов;

- **Merge** – объединение сигналов в вектор;
- **Switch** – ручной переключатель сигналов;
- **Selector** – селектор элементов из векторов или матриц;
- **From** – прием сигнала от блока **Goto**;
- **Goto Tag Visibility** – блок признака видимости сигнала;
- **Goto** – организация «беспроводного» передатчика данных.

В разделе **Signal Storage & Access** имеются 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.

5.7.2. Блок создания шины Bus Creator

Блок создания шины **Bus Creator** служит для объединения двух и более сигналов самого различного вида в одну шину. Применение этого блока иллюстрирует рис. 5.39. Здесь два сигнала (синусоидальный и прямоугольные импульсы) объединены в одну шину, и оба этих сигнала показывает осциллограф Scope.

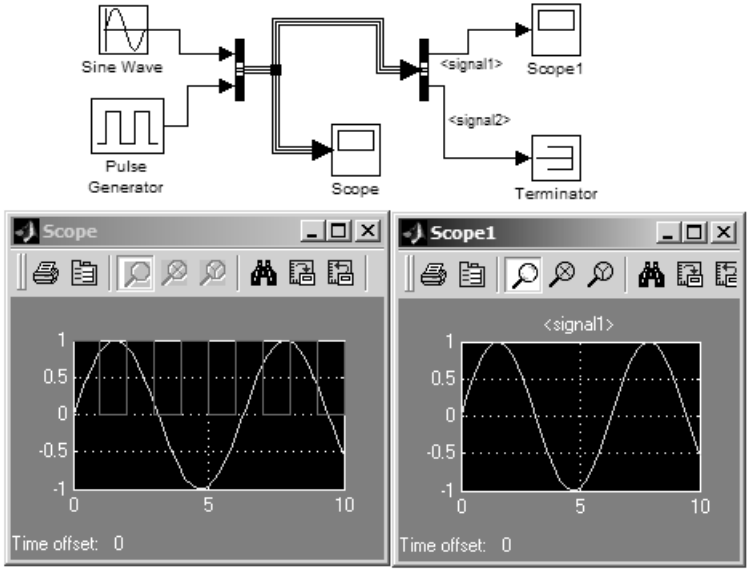


Рис. 5.39. Пример работы с шиной

Окно параметров блока **Bus Creator** показано на рис. 5.40. Работа с ним вполне очевидна.

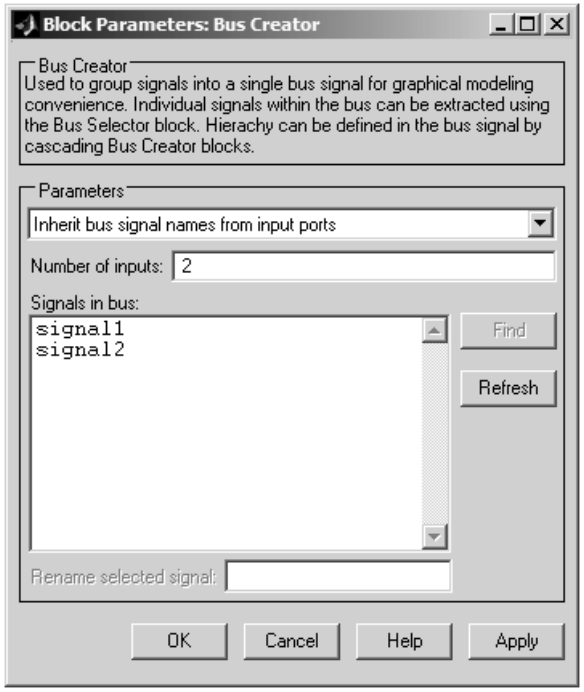


*Рис. 5.40. Окно параметров блока **Bus Creator***

5.7.3. Блок шинного селектора Bus Selector

Блок шинного селектора **Bus Selector** обеспечивает выбор заданных сигналов из нескольких. В примере, приведенном на рис. 5.39, блок шинного селектора выделяет оба сигнала: один из них направляется на осциллограф Scope1, а другой поглощается заглушкой. В итоге на этом осциллографе виден лишь один из сигналов – синусоидальный.

Окно параметров шинного селектора представлено на рис. 5.41. 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.

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** дает возможность 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.

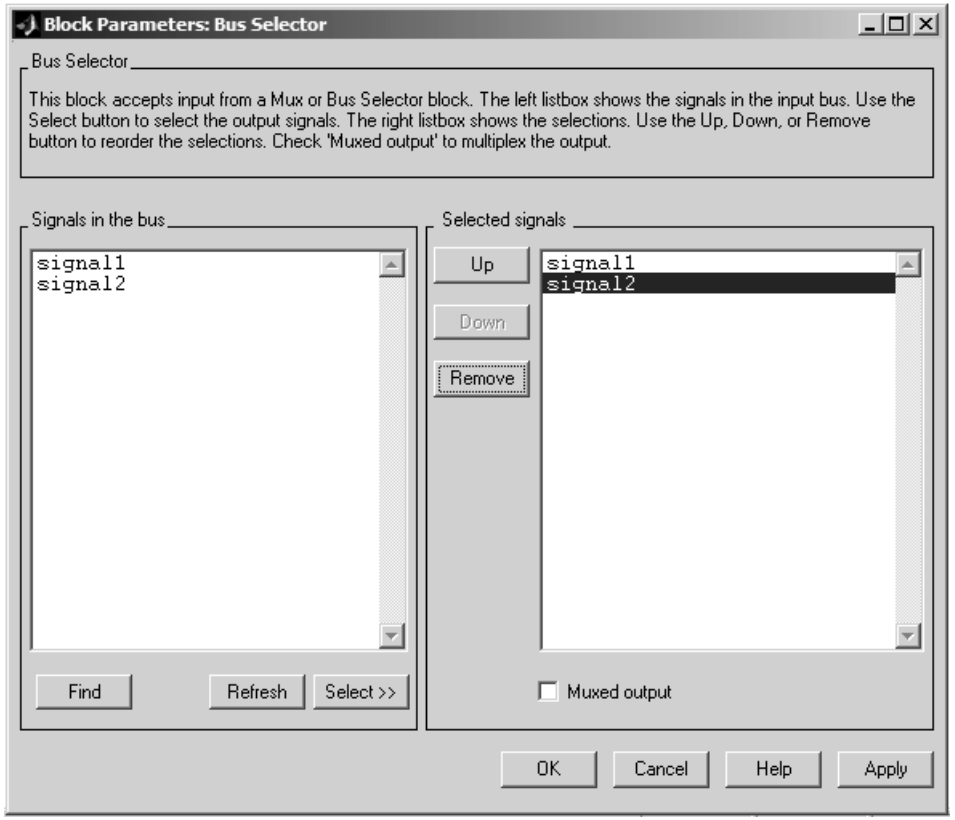


Рис. 5.41. Окно установки параметров шинного селектора

5.7.4. Блок выбора последнего сигнала Merge

Блок **Merge** служит для объединения двух и более векторных сигналов в один векторный сигнал. Пример применения блока для объединения обрезанного одного вектора с другим вектором представлен на рис. 5.42.

Также показано окно установки параметров блока **Merge**. Можно задать:

- число входов **Number if inputs**;
- начальное значение выходного сигнала **Initial output** (если параметр не задан, возвращается сигнал, заданный последним);
- разрешение на различную размерность входных портов **Allow unequal port width**;
- смещение входного сигнала **Input Port offset**.

С помощью параметра **Input Port offset** можно задавать расположение входного сигнала в выходном векторе.

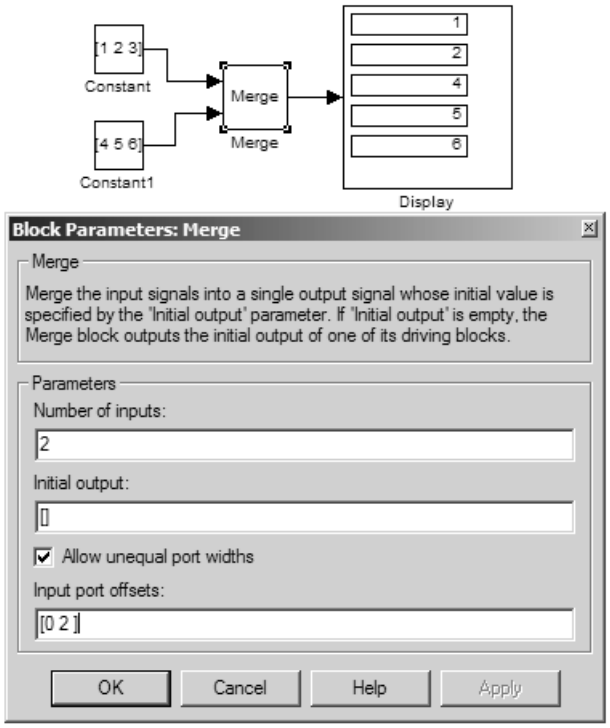


*Рис. 5.42. Применение блока **Merge** для объединения двух векторов*

Еще один пример mergedemo представлен на рис. 5.43. Здесь блок **Merge** создает сигнал, объединяющий поочередные вырезки синусоидального сигнала и пилообразного сигнала.

5.7.5. Блок мультиплексирования Mux

Блок **Mux** предназначен для мультиплексирования (объединения) сигналов. Их действие демонстрирует рис. 5.44. 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. 5.44) устанавливается число входов и один из трех вариантов изображения блока.

5.7.6. Блок демультиплексирования Demux

Обратная задача – разделение сигналов – решается блоком демультиплексирования **Demux**. Применение этого блока также показано на рис. 5.44. Там же показано окно параметров блока демультиплексирования.

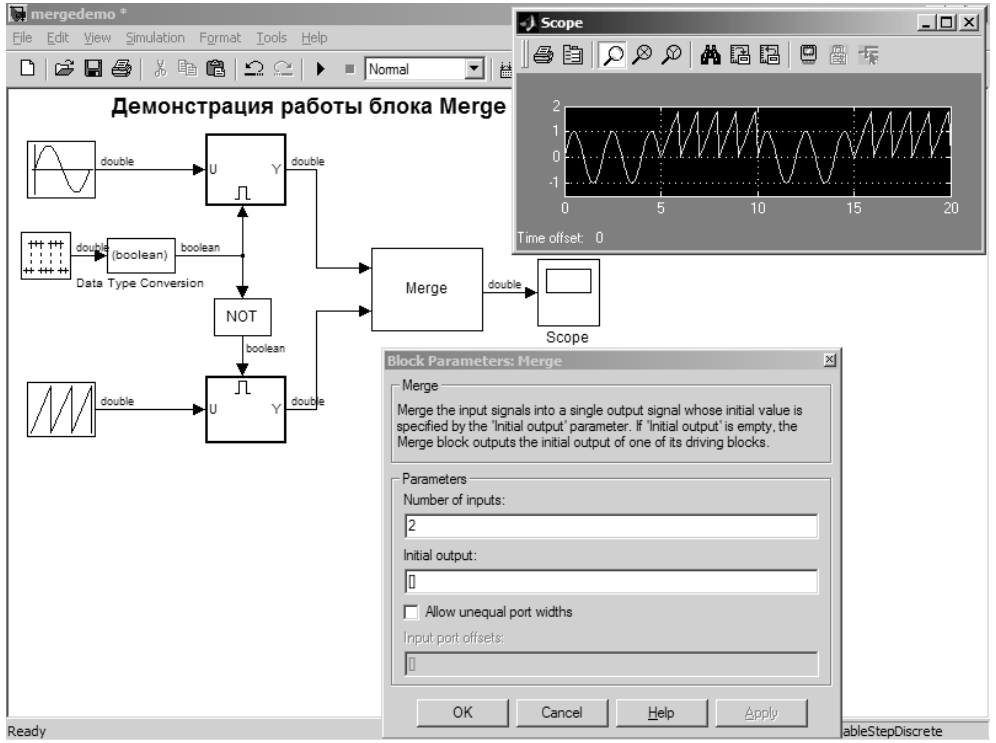


*Рис. 5.43. Применение блока **Merge** для объединения сигналов*

5.7.7. Блоки для записи и считывания данных 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 тремя блоками.

Блок **Data Store Memory** служит для записи поименованных данных в память, то есть он по существу резервирует под данные память. Можно записывать (**Data Store Write**) и считывать (**Data Store Read**) поименованные данные. Рисунок 5.45 показывает применение этих блоков для записи в память сигнала с постепенно нарастающей частотой и затем считывания его и отображения осциллографом.

На рис. 5.45 показаны также окна параметров блоков записи данных в память и считывания их из памяти. Параметры блоков вполне очевидны. Рисунок 5.46 показывает окно параметров блока **Data Store Memory**.

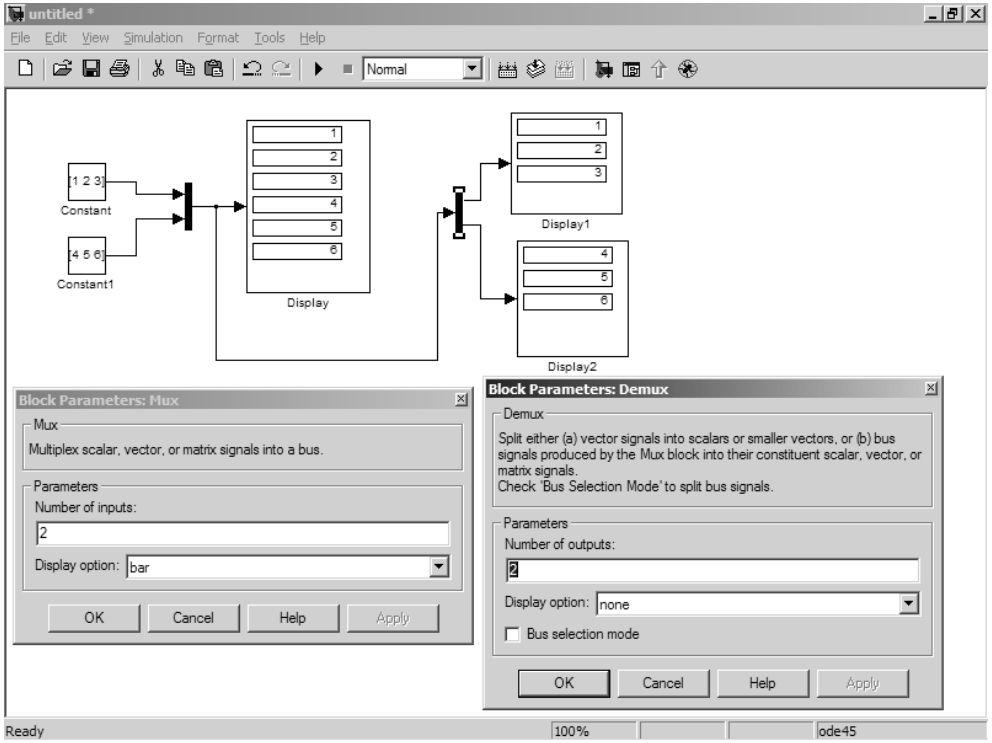


*Рис. 5.44. Применение блока **Mux***

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 (к примеру, если этот параметр задан как [[1 2] [3 4]], то это означает, что он является матрицей размера 2×2);
- `Interpret vector parameters as 1-D` – опция, задающая 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**. Блок **Data Store Memory** может обслуживать несколько блоков **Data Store Write** и **Data Store Read**. Его расположение в диаграмме модели задает область видимости сохраняемых данных (в частности, переменных). Если блок **Data Store Memory** размещен в диаграмме верхнего уровня, то все ее данные будут «видимы», то есть доступными для сохранения или считывания. Если же блок **Data Store Memory** расположен в подмодели, то доступными будут данные только этой подмодели.

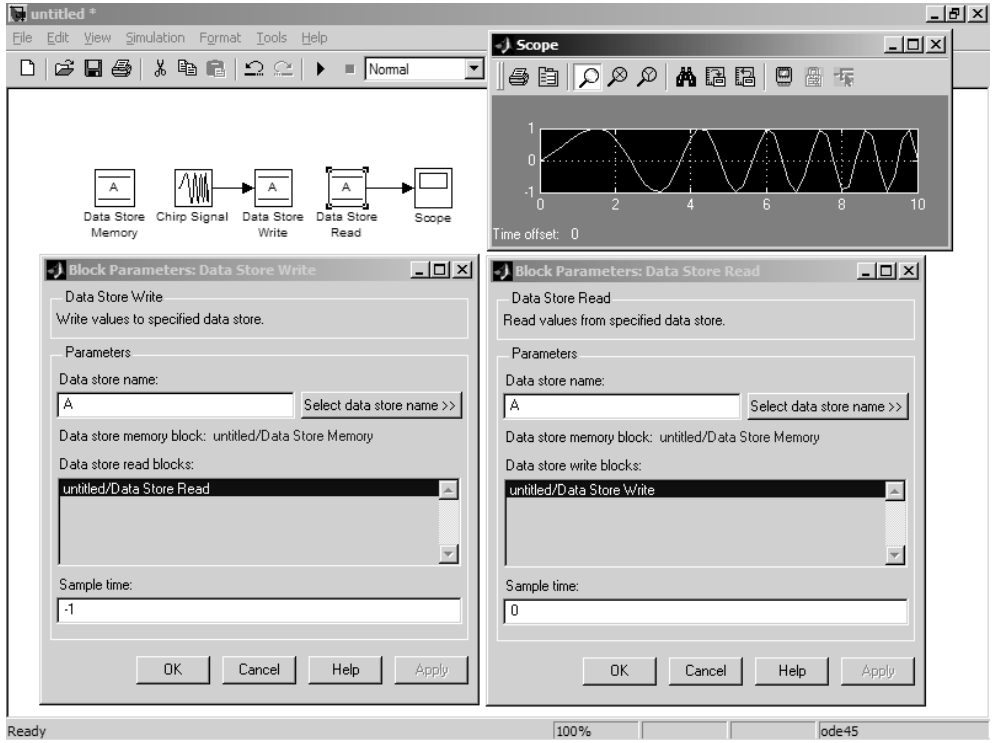


Рис. 5.45. Применение блоков задания области памяти, записи в нее данных и их считывания

5.7.8. Блоки «беспроводной» связи 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. Применение данных блоков иллюстрирует рис. 5.47. Там же показаны окна настройки этих блоков.

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 видимых прямых соединений между ними и блоком **Goto** нет.

Окно блока, помимо параметра **Tag** (имени сигнала), имеет параметр **Tag Visibility** (признак видимости), который может принимать одно из трех значений:

- **local** – передача сигнала в пределах данной подсистемы (имя сигнала задается в квадратных скобках, например [A]);

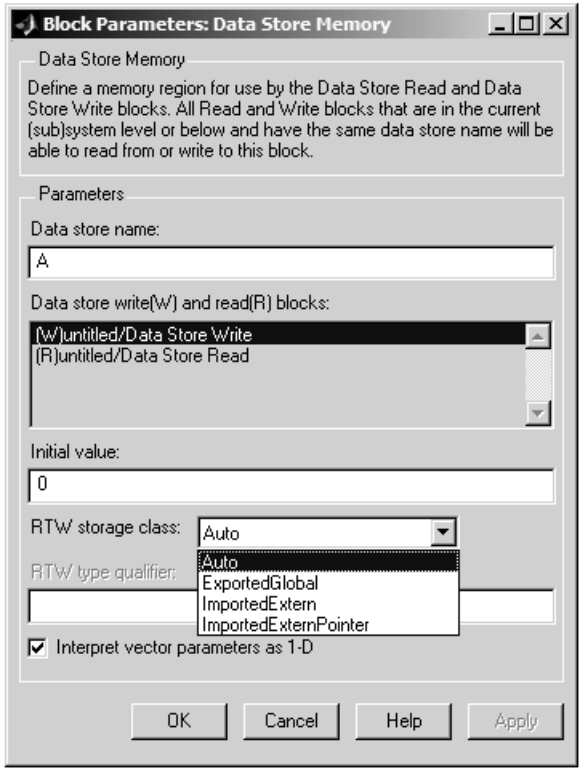


Рис. 5.46. Окно параметров блока ***Data Store Memory***

- **scoped** – передача сигнала в пределах данной подсистемы и подсистем более низкого уровня (имя сигнала задается в фигурных скобках, например {A});
- **global** – передача сигнала в пределах всей модели (имя сигнала непосредственно, например A).

В окне параметров блока `From` устанавливается единственный параметр – имя сигнала. Для этого служит список, управляемый кнопкой `Select tag>>`. От одного блока `Goto` может работать множество блоков `From`. Правила видимости сигналов в блоках `From` те же, что и у блоков `Goto`. Блок **Goto Tag Visibility** служит для установки видимости в пределах модели или подсистемы, на которую распространяется область видимости. В окне его параметров устанавливается только имя сигнала.

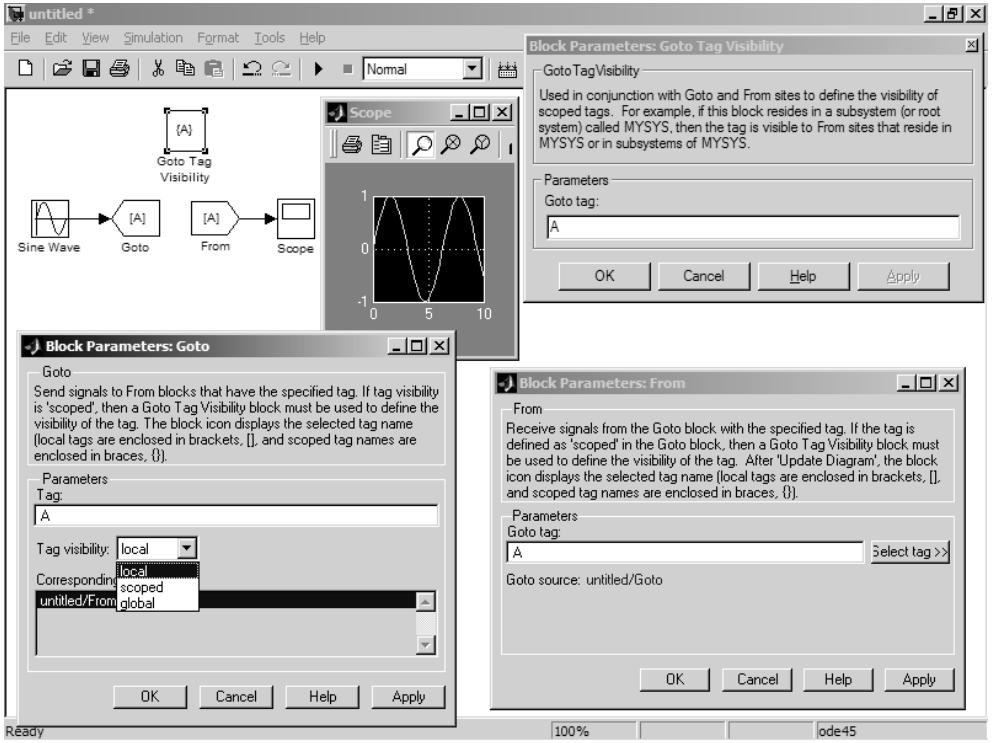


Рис. 5.47. Применение блоков «беспроводной» связи

5.7.9. Ручной переключатель сигналов Manual Switch

Ручной переключатель сигналов служит для подключения одного из двух входных сигналов на выход при ручном управлении – двойном щелчке левой клавиши мыши на пиктограмме переключателя. Переключение возможно как в процессе моделирования, так и при его остановке. Окна параметров блок не имеет. Пример применения блока ручного переключателя дан на рис. 5.48. Показаны два варианта положения переключателя, обеспечивающего переключение сигналов, – прямоугольных импульсов и синусоидального.

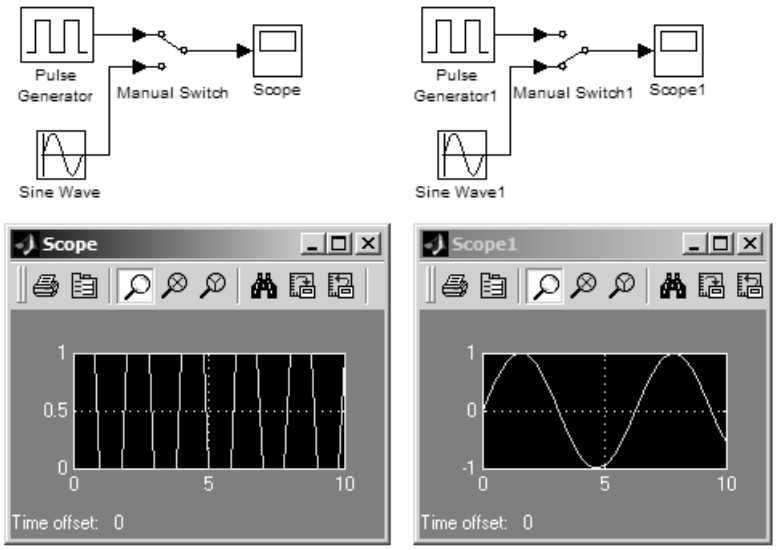


Рис. 5.48. Применение ручного переключателя сигналов

5.7.10. Управляемый переключатель сигналов Switch

Управляемый переключатель **Switch** подобен ручному, но управляется управляющим сигналом – рис. 5.49. Здесь показаны два варианта применения переключателя при сигнале, превышающем нулевой порог и меньшем его. При этом переключатель передает на выход любо импульсный сигнал, либо синусоидальный сигнал.

Окно параметров переключателя, показанное на рис. 5.50, позволяет из списка критериев (на рисунке он открыт) переключения задать нужный критерий и численное значение порога переключения. Опция **Show additional parameter** расширяет окно и позволяет установить ряд дополнительных параметров: установку идентичности всех входных данных, установку типа входных данных, способ округления чисел в данных, корректность ограничения целочисленных данных и фиксацию прохождения сигналов через нуль.

Блок **Switch** удобно использовать для создания вырезок из сигнала (рис. 5.51) или сигналов, состоящих из сшитых кусков различных сигналов. Для переключения сигналов при этом используется управляющий сигнал в виде прямоугольных импульсов.

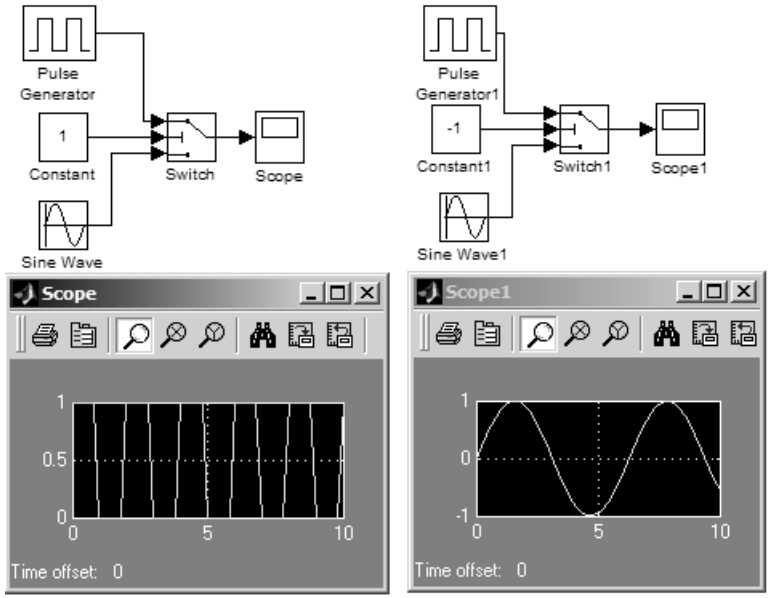


Рис. 5.49. Применение управляемого переключателя сигналов

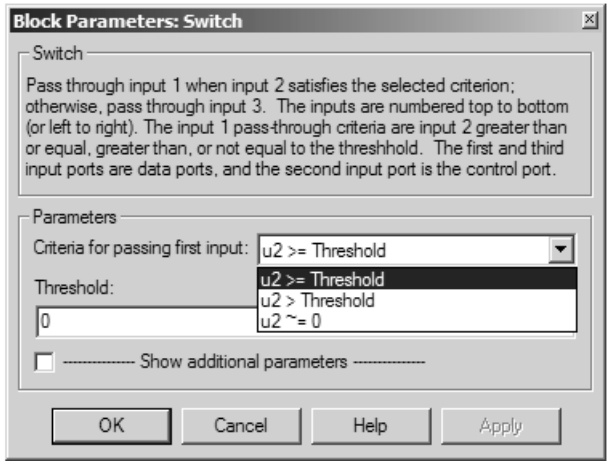


*Рис. 5.50. Окно установки параметров блока **Switch***

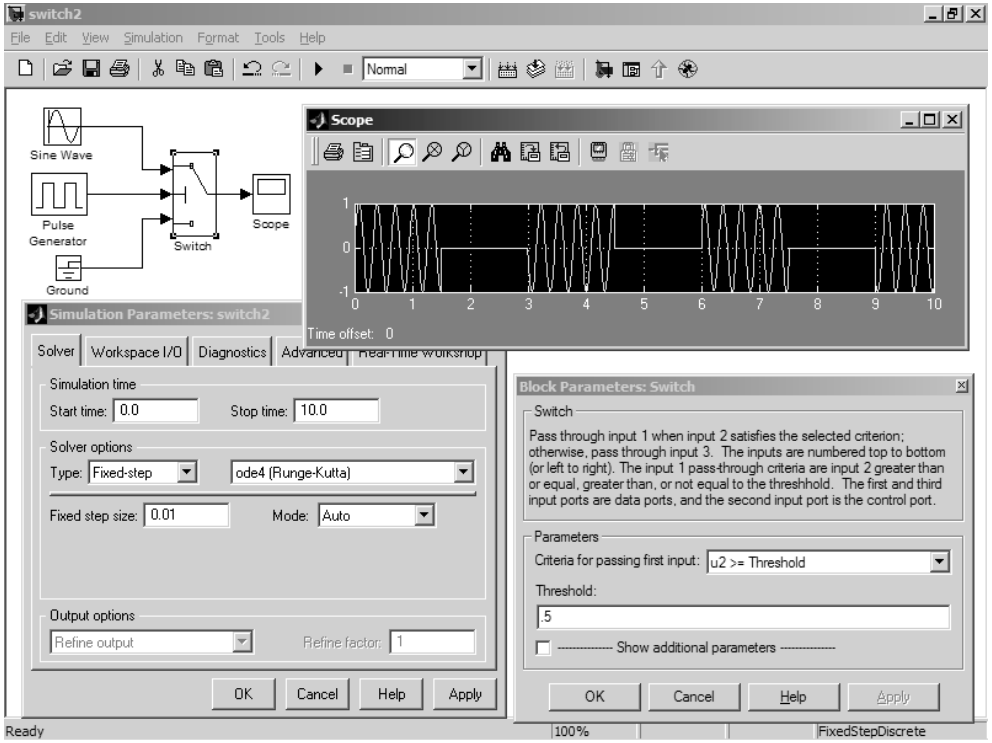


*Рис. 5.51. Применение блока **Switch** для формирования пачек синусоиды*

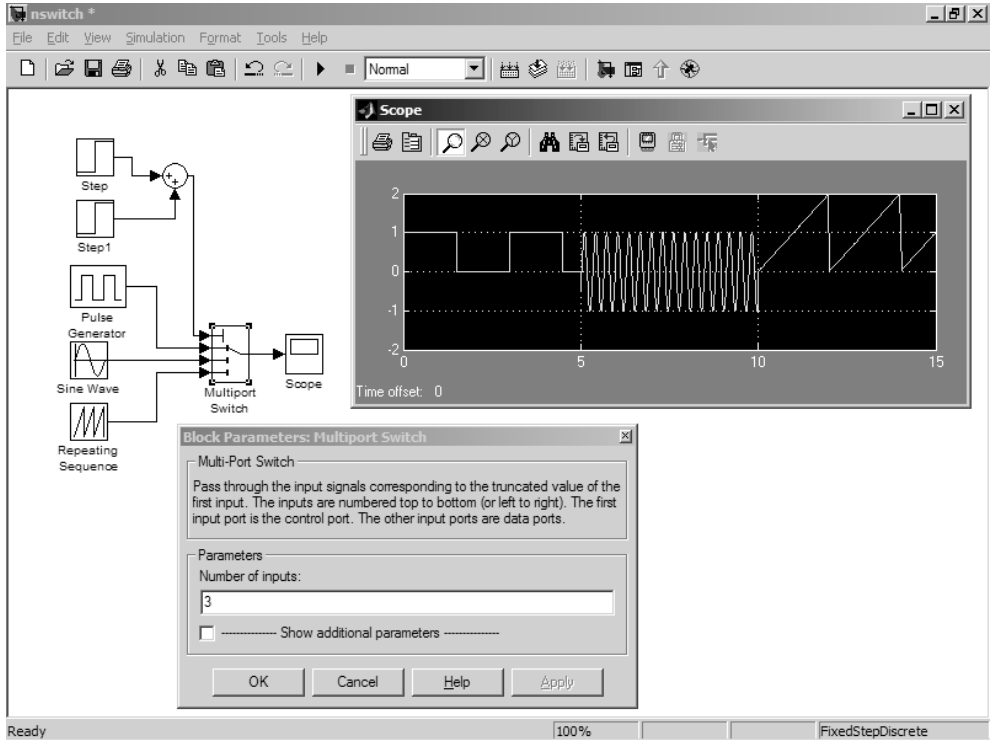


*Рис. 5.52. Применение блока **Multiport Switch** для переключения трех сигналов*

5.7.11. Многопортовый переключатель сигналов Multiport Switch

Многопортовый переключатель **Multiport Switch** обеспечивает коммутацию одного из ряда сигналов на выход под действием управляющего сигнала, указывающего номер входного порта (1, 2, 3 и т. д.). Переключение возможно в динамике процесса моделирования, что позволяет формировать сложные последовательности сигналов.

Наглядный пример применения переключателя **Multiport Switch** представлен на рис. 5.52. С помощью блоков **Step** формируется сигнал, имеющий значения 1, 2 и 3 на отрезках времени 5 при общем времени 5. Этот сигнал подается на управляющий вход и обеспечивает последовательную передачу на выход блока сигналов прямоугольной формы, синусоиды и пилы.

Окно установки параметров блока **Multiport Switch** в упрощенном виде также представлено на рис. 5.52. В нем устанавливается только число входов (портов) блока. Задание опции **Show additional parameter** расширяет окно (рис. 5.53) и позволяет задать ряд дополнительных параметров. Назначение дополнительных параметров указывалось при описании блока **Switch**.

5.7.12. Селектор Selector

Блок селектора **Selector** предназначен для выборки из векторов или матриц заданных элементов. Три простых его применения представлены на рис. 5.54. Там же показаны окна задания параметров этого блока для первого и третьего примеров. Нетрудно увидеть, что изображение блока зависит от того, какой выбор элементов задан.

Выбор сигнала может быть внутренним (internal) или внешним (external). В последнем случае у блока появляются выводы для управляющих сигналов (второй и третий примеры рис. 5.54).

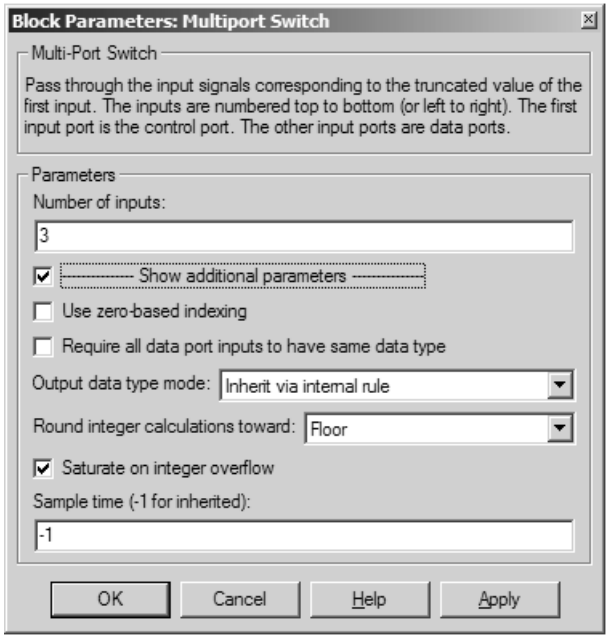


*Рис. 5.53. Окно установки параметров блока **Multiport Switch** в расширенном варианте*

5.8. Библиотека атрибутов сигналов Signal Attribute

5.8.1. Состав библиотеки атрибутов сигналов

Библиотека атрибутов сигналов в пакете Simulink 5 содержит всего 6 блоков, разбитых на два раздела: **Signal Attribute Manipulation** и **Signal Attribute Detection**. Окно этого раздела библиотеки показано на рис. 5.55.

5.8.2. Блок преобразования типов сигналов Data Type Conversion

Блок **Data Type Conversion** служит для преобразования типов данных. Для векторов и матриц такое преобразование может иметь единственный заданный тип. Вполне очевидные примеры применения этого блока представлены на рис. 5.56. Там же дано окно установки параметров блока.

В окне параметров блока из списка (на рис. 5.56 он открыт) можно выбрать тип данных для выхода блока. Кроме того, можно задать опцию **Saturate on integer overflow**, при которой подавляется переполнение сигналов целочисленного типа.

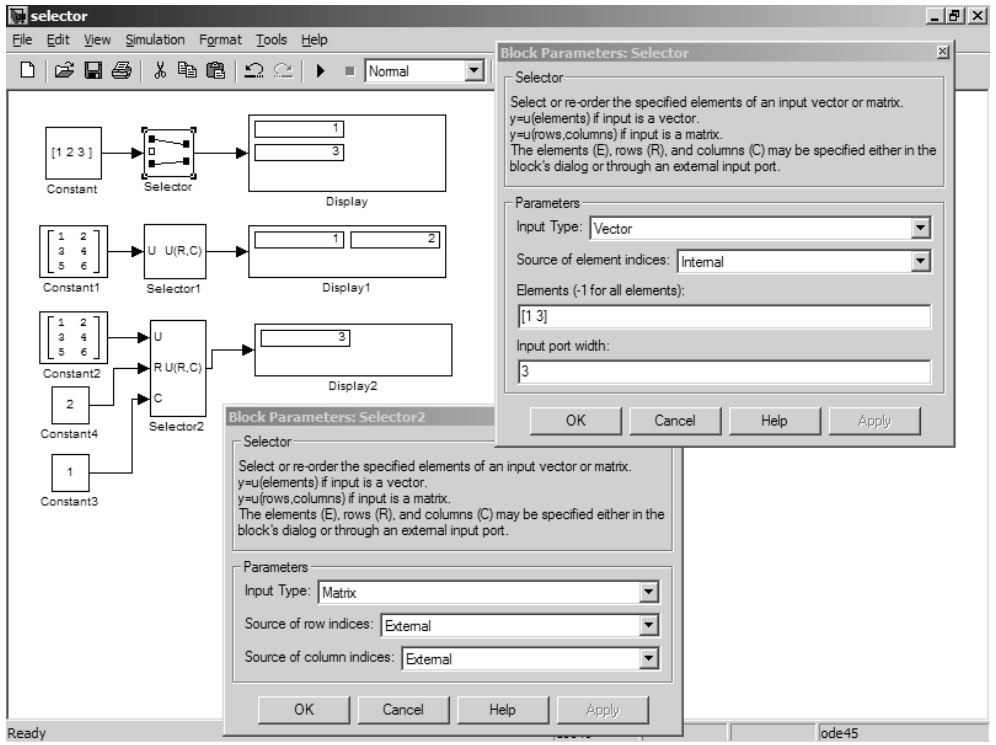


*Рис. 5.54. Примеры применения блока селектора **Selector***

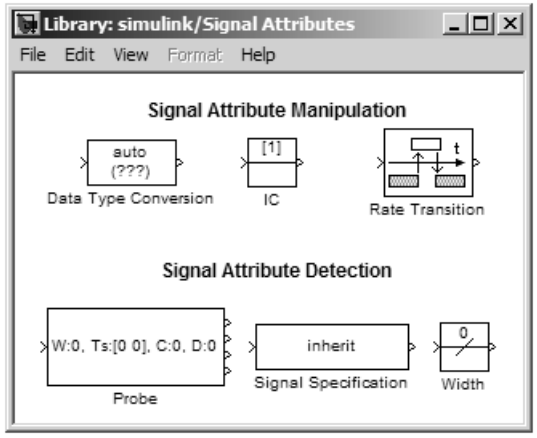


Рис. 5.55. Окно библиотеки атрибутов сигналов

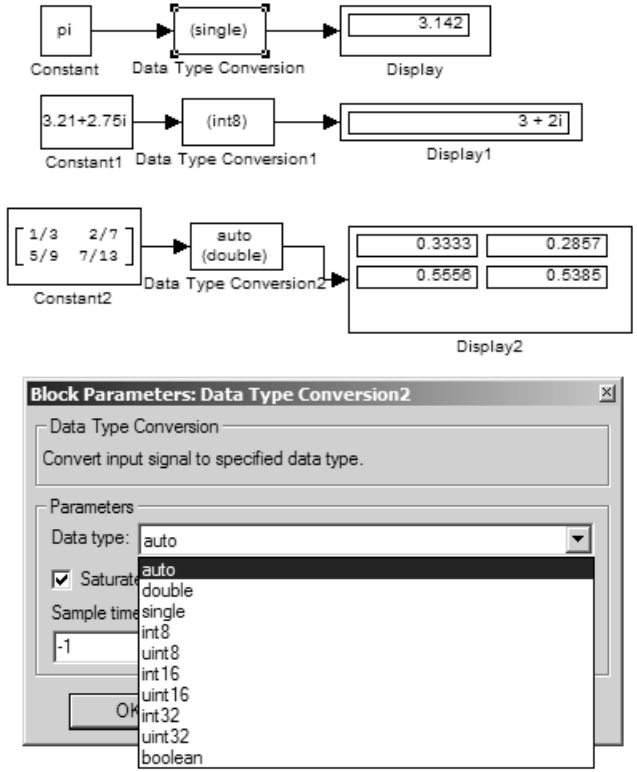


Рис. 5.56. Применение блока ***Data Type Conversion***

5.8.3. Установка начального значения сигнала IC

Блок **IC** служит для задания начального значения сигнала (Initial Condition). Применение блока поясняет рис. 5.57. Используется синусоидальный сигнал с естественным начальным значением, равным 0, который с помощью блока **IC** сделан равным 1. Последующие значения сигнала не изменяются. В окне установки параметров блока устанавливается начальное значение сигнала и параметр **Sample time**.

5.8.4. Блок согласования дискретных значений Rate Transition

Блок согласования дискретных значений **Rate Transition** используется для согласования дискретных блоков, имеющих разные периоды квантования. Окно параметров этого блока представлено на рис. 5.58.

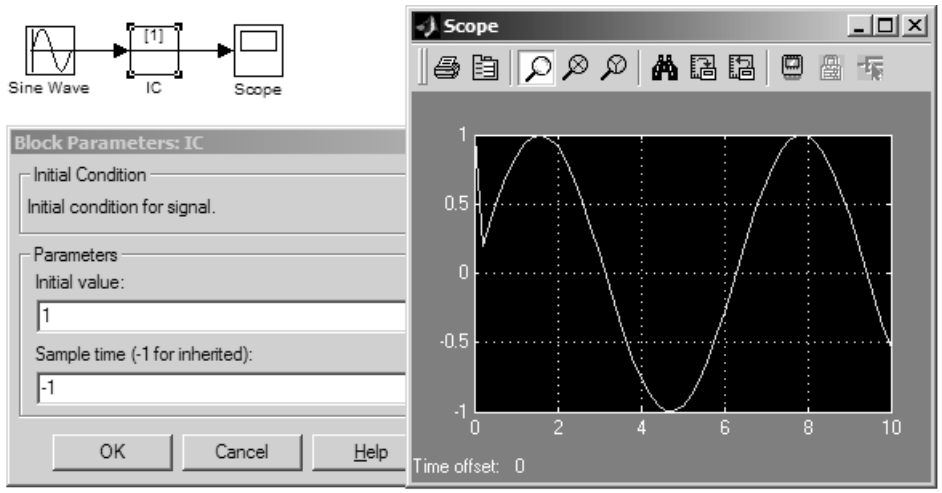


Рис. 5.57. Применение блока ***IC***

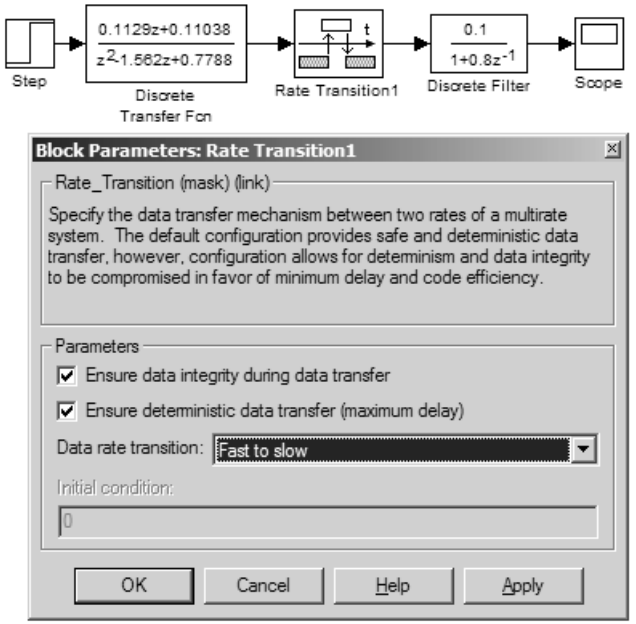


Рис. 5.58. Блок ***Rate Transition*** *и окно его параметров*

В окне настроек можно установить опции гарантированной целостности данных и детерминированной передачи их. Наиболее важны установки **Data rate transition** (режима передачи данных). Возможна одна из установок:

- **Slow to fast** – передача от медленного блока к быстрому;
- **Fast to slow** – передача от быстрого блока к медленному.

Есть также опция задания начального значения **Initial condition**.

5.8.5. Блок спецификации сигнала Signal Specification

Блок спецификации сигнала **Signal Specification** служит для распознавания сигналов на его входе и выдачи спецификации сигнала. Пример применения блока дан на рис. 5.59.

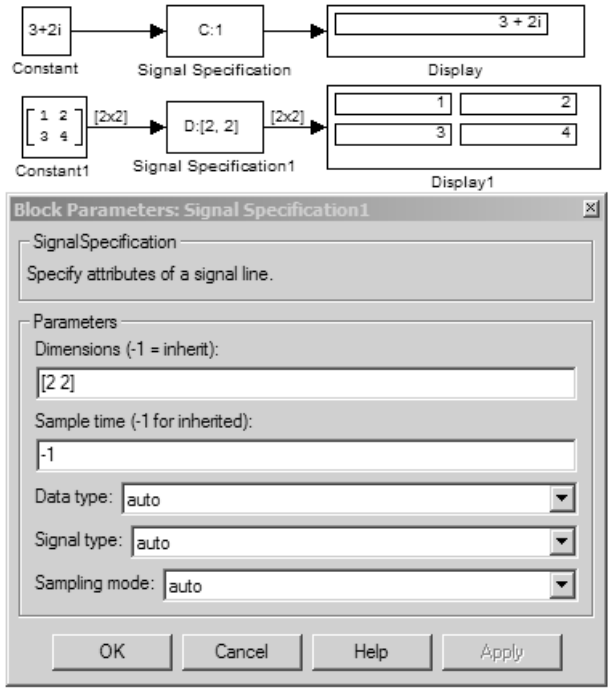


Рис. 5.59. Применение блока ***Signal Specification***

В окне параметров блока устанавливается размерность сигнала **Dimension** (в виде [m n] или –1 для любой размерности, эталонное время **Sample time**, тип данных **Data Type**, тип выходных данных **Output data type**, тип сигнала **Signal type** и режим **Simpling Mode**.

5.8.6. Блок проверки сигналов Probe

Блок **Probe** осуществляет более детальную проверку сигналов, чем блок **Signal Specification** (рис. 5.60). На этом рисунке представлены также окна параметров источника синусоидального сигнала и блока **Probe**. При возникновении ошибки с выходов блока **Probe** считываются значения тех параметров, которые указаны в окнах установки их параметров. Их можно использовать для оценки ситуаций, вызывающих ошибки и их устранения в ходе моделирования.

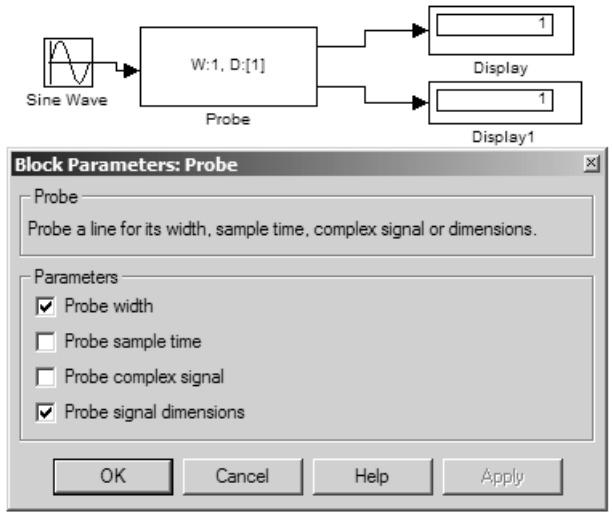


Рис. 5.60. Применение блока ***Signal Specification***

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.

5.8.7. Блок вычисления размера сигнала Width

Для вычисления полного размера (длины) сигнала используется блок **Width**. Примеры применения его и окно установки параметров показаны на рис. 5.61. При установке расширенного окна можно устанавливать тип выходного сигнала с помощью двух списков.

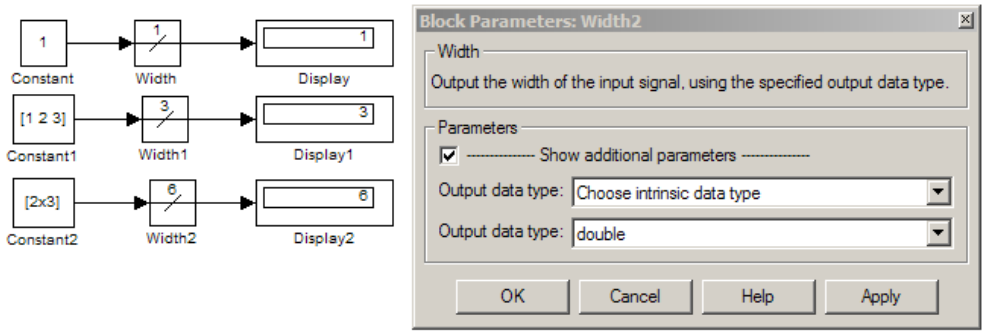


Рис. 5.61. Применение блока ***Signal Specification***

5.9. Новые источники в Simulink 6.6

5.9.1. Окно источников сигналов Simulink 6.6

Окно источников сигналов библиотеки блоков Simulink 6.6 показано на рис. 5.62. Оно отличается от окна Simulink 5.1 (рис. 5.1) наличием четырех новых источников. Все они описаны ниже и относятся к источникам генерации различных последовательностей.

5.9.2. Источник Repeating Sequence Stair

Источник **Repeating Sequence Stair** служит для имитации повторяющихся ступенчатых последовательностей. На рис. 5.63 представлена простейшая модель источник–осциллограф для этого источника и окно осциллографа с осциллограммой сигнала от него.

В блоке задания параметров этого источника задаются всего два параметра – вектор уровней ступенек и время Sampling time. По умолчанию заданы вектор [3 1 4 2 1].' (поскольку удобно задавать вектор-строку, то для задания вектора уровней записана операция транспонирования) и время Sampling time, равное –1.

5.9.3. Источник Repeating Sequence Interpolated

Источник **Repeating Sequence Interpolated** предназначен для имитации последовательностей с заданными уровнями в узловых точках и интерполяцией значений между ними. В панели параметров этого блока задаются два вектора – ординат (уровней) узловых точек и моментов времени для них (–1). Можно также задать один из четырех типов интерполяции и время Sampling Time. Пример применения и контроля выхода блока показан на рис. 5.63.

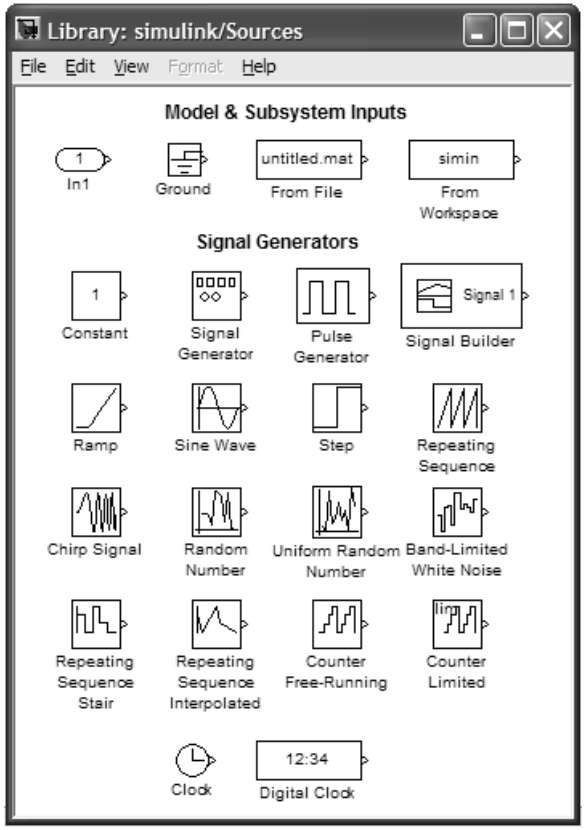


Рис. 5.62. Применение блока ***Signal Specification***

5.9.4. Источник Counter Free-Running

Источник **Counter Free-Running** служит для генерации последовательности отсчетов дискретного времени с инициализацией по нулевому уровню. В панели параметров этого блока задается число бит последовательности (по умолчанию 16) и время Sampling Time (–1). Пример работы блока дан на рис. 5.63.

5.9.5. Источник Counter Limited

Источник **Counter Limited** генерирует ограниченную последовательность отсчетов дискретного времени с инициализацией по нулевому уровню. В окне параметров задаются верхний предел Uper Limit (по умолчанию 7) и время Sampling Time (–1). Пример работы блока дан на рис. 5.63.

Набор получателей и регистраторов сигналов в Somulink 6.6 изменений не претерпел. Окно с ними соответствует показанному на рис. 5.27.

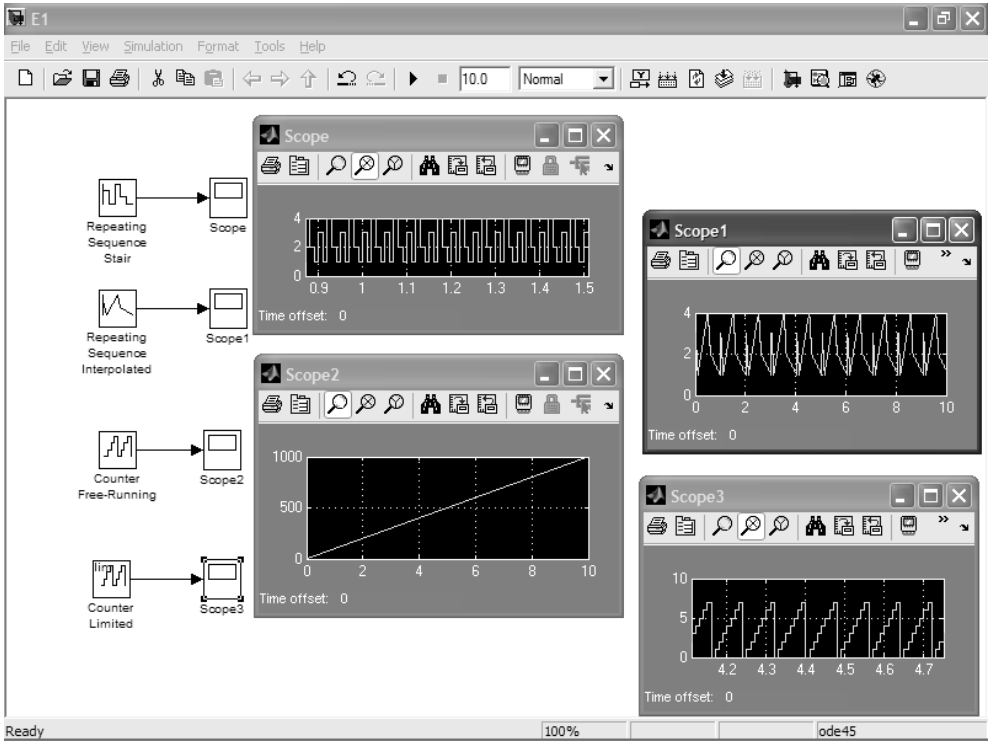


Рис. 5.63. Примеры применения новых блоков источников Simulink 6.6

Урок 6

Математические блоки

В процессе моделирования различных систем и устройств, а также при создании новых блоков часто требуется выполнение различных математических операций. Для этого Simulink обладает набором блоков математических операций, входящих в библиотеку Math. С ними мы познакомимся в данном уроке.

6.1. Математическая библиотека Math

6.1.1. Обзор библиотеки Math

Большие возможности в моделировании различных систем предоставляет библиотека математических блоков **Math**. Окно библиотеки **Math** показано на рис. 6.1.

Библиотека математических блоков в Simulink 5 разбита на четыре раздела:

- `Math Operations` – математические операции;
- `Vector Operations` – векторные операции;

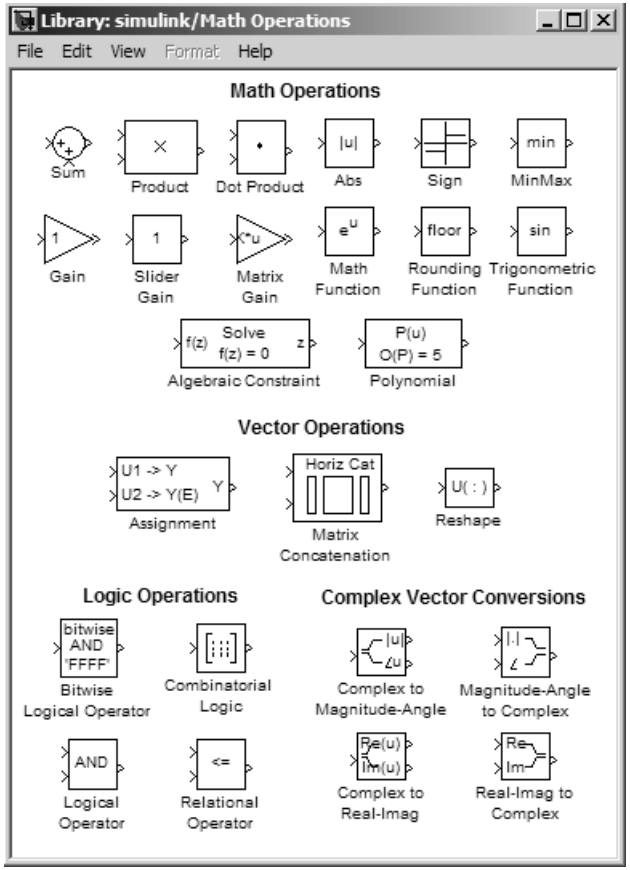


Рис. 6.1. Окно библиотеки математических блоков Simulink 5.1

- `Logic` – логические операции;
- `Complex Vector Conversion` – преобразование векторов с комплексными элементами.

Ввиду очевидности англоязычных названий математических блоков в перечислении их нет особого смысла. 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.

6.1.2. Блоки выполнения арифметических операций

К числу наиболее простых математических блоков относятся блоки арифметических операций: вычисления абсолютного значения числа **Abs**, знака числа **Sign**, округления `Rounding Function`, скалярного произведения **Dot Product**, обычного произведения **Product**, а также суммы **Sum**. Рисунок 6.2 показывает применение ряда из этих блоков. Там же даны окна установки параметров для двух из блоков.

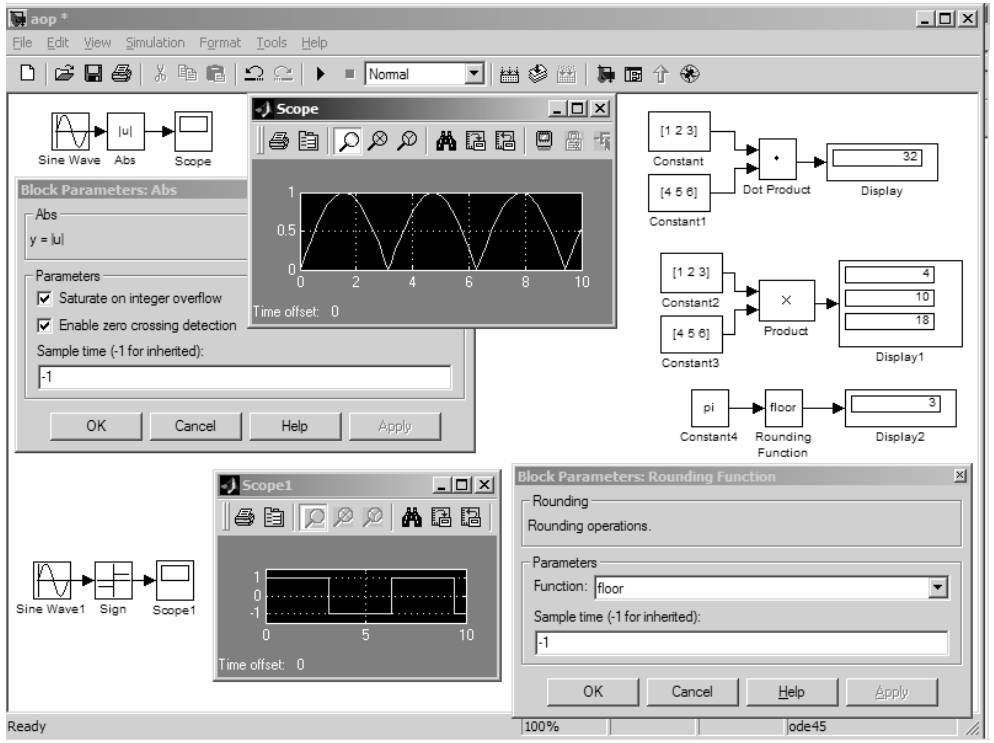


Рис. 6.2. Примеры применения блоков арифметических операций

Из этих примеров полезно особо отметить применение блока **Abs** для моделирования двухполупериодного выпрямления синусоидального сигнала и блока `Sign` для формирования меандра из синусоиды. Блок **Product** наряду с умножением скалярных сигналов может использоваться и для вычисления произведения векторов (пример на рис. 6.2) и матриц. В окне параметров этого блока можно задать число его входов, то есть блок можно использовать и при числе сомножителей более 2.

Блок функций округления `Rounding Function` позволяет в окне установки параметров выбрать из списка одну из четырех функций округления:

- `floor` – до ближайшего меньшего целого;
- `ceil` – до ближайшего большего целого;
- `round` – до ближайшего целого;
- `fix` – до целого, полученного отбрасыванием целой части.

При векторном и матричном сигналах блок обеспечивает округление поэлементно. Могут использоваться скалярные и комплексные сигналы.

Это относится и к блоку суммирования, примеры применения которого представлены на рис. 6.3. 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.

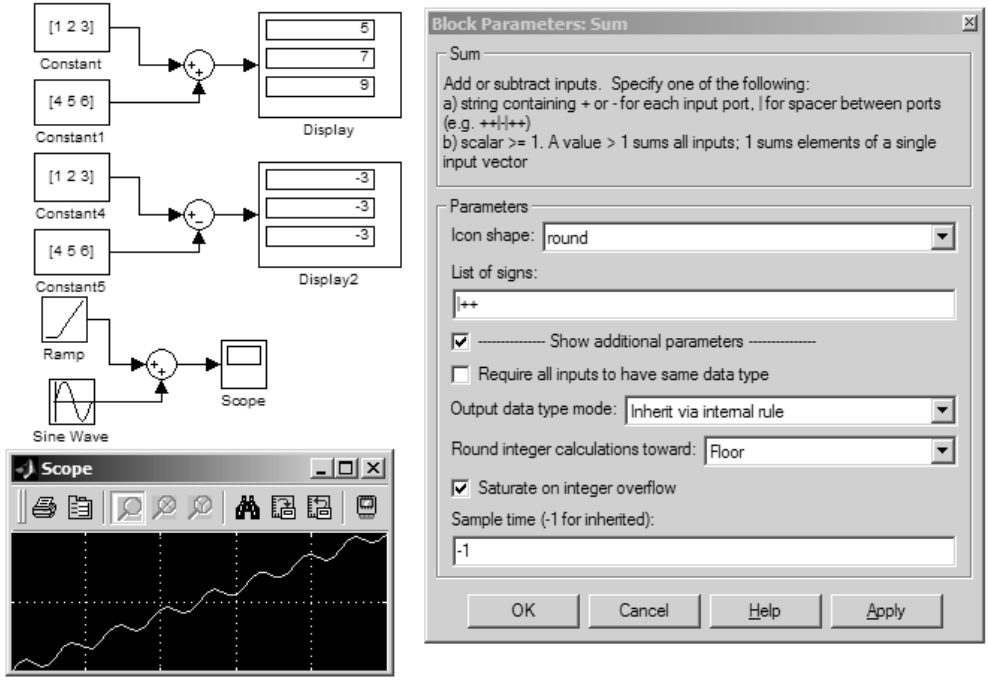


Рис. 6.3. Примеры применения блока суммирования

Блок **Product (Умножение)** предназначен не только для умножения, но и для деления. 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.

6.1.3. Блоки вычисления элементарных функций

На рис. 6.4 представлены три блока, выполняющих вычисления математических функций: **Math Function (Математическая функция)**, **Trigonometric Function (Тригонометрическая функция)** и **Rounding Function (Функция округления)**.

Действие этих функций и установки их параметров достаточно очевидны. Выбор конкретной вычисляемой функции осуществляется в раскрывающемся списке – рис. 6.4. В нем имеется типовой набор элементарных функций.

6.1.4. Блок выполнения логических операций Logical Operation

Блок `Logical Operation` позволяет задавать любую из известных базовых логических операций. Ввиду общеизвестности этих операций ограничимся примером выполнения операций `AND` и **OR**, представленным на рис. 6.5. Пример формирует таблицы истинности для этих операций.

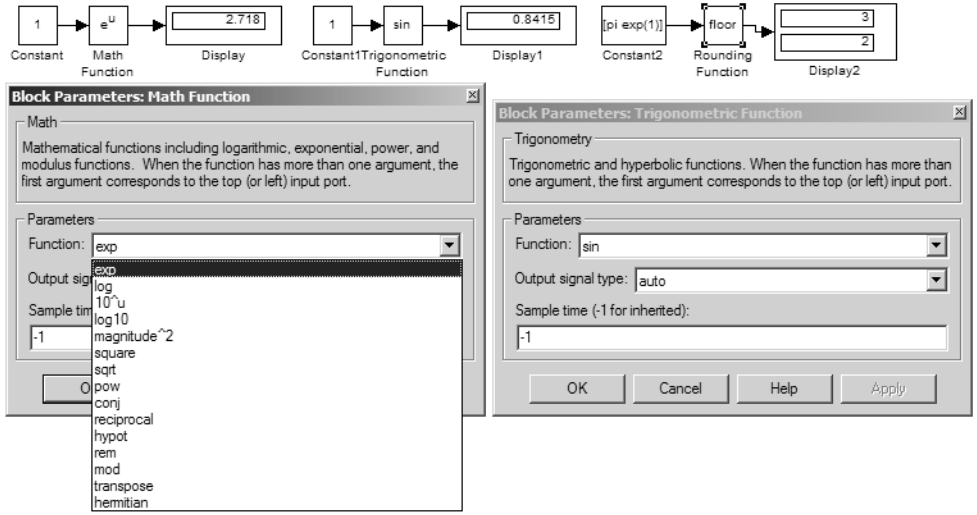


Рис. 6.4. Примеры применения блоков арифметических операций

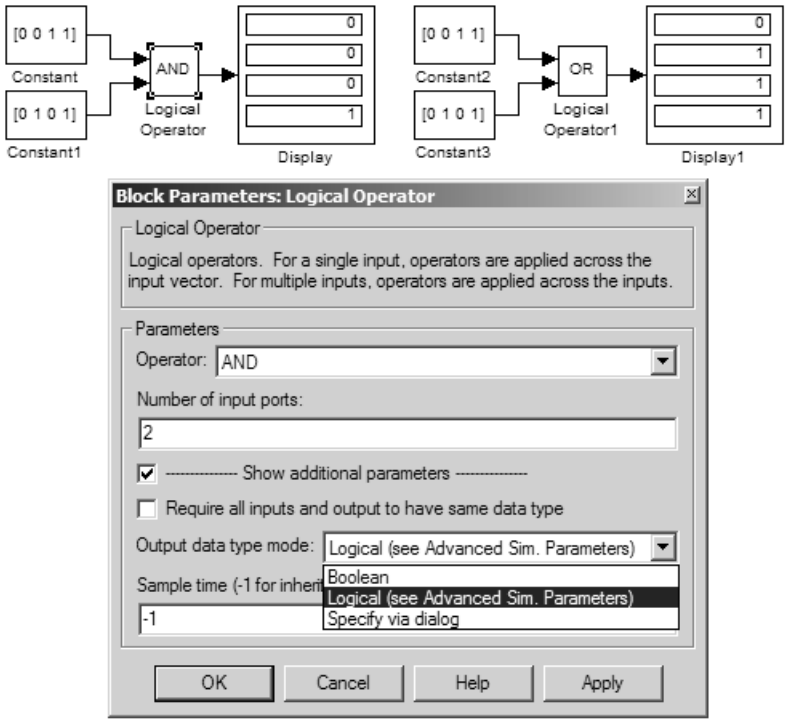


*Рис. 6.5. Пример выполнения операции **AND** и **OR***

На рис. 6.5 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).

Обратите внимание, что в приведенном примере входными данными являются векторы с логическими сигналами – логическим нулем 0 и логической единицей 1. В расширенном окне установки параметров можно задать режим вывода типа выходных данных и параметр Simple time.

6.1.5. Блок выполнения побитовых логических операций `Bitwise Logical Operator`

Для выполнения побитовых операций над данными в формате unit8, unit16 и utit32 служит блок `Bitwise Logical Operator`. Пример его применения и окно установки параметров блока показаны на рис. 6.6.

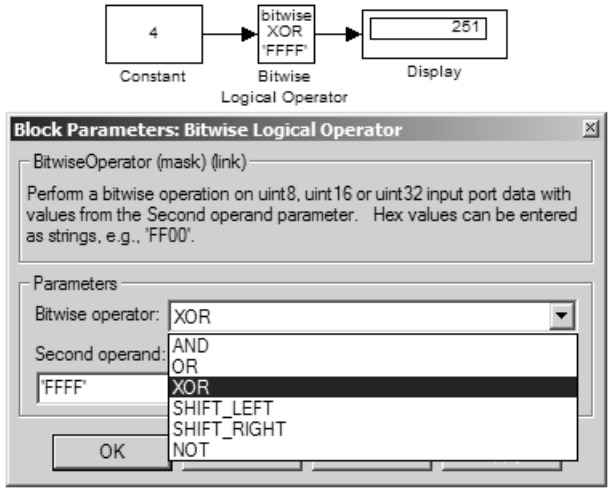


*Рис. 6.6. Пример выполнения побитовой операции **XOR***

6.1.6. Блок выполнения операций по таблице истинности `Combinatorial Logic`

Блок `Combinatorial Logic` служит для реализации логических операций по заданной таблице истинности, широко применяемой в теории автоматов. Таблица истинности – это список возможных логических состояний блока. Число ее строк $r = 2^c$, где c – число входных сигналов u (при числе входных сигналов $m = 2$ имеем 4 строки). Индекс каждой строки определяется входными сигналами:

$$r\_index = 1+u(m)*2^0+u(m-1)*2^1+...+u(1)*2^{m-1}.$$

Ниже дано построение таблицы истинности для двух сигналов и логической операции И (AND)^.

Вход 1	Вход 2	Выражение для r_index	Значение r_index	Выход
0	0	$1+0*2^0+0*2^1$	1	0
1	0	$1+1*2^0+0*2^1$	2	0
0	1	$1+0*2^0+1*2^1$	3	0
1	1	$1+1*2^0+1*2^1$	4	1

На рис. 6.7 показан пример применения блока `Combinatorial Logic` для выполнения данной операции в соответствии с созданной таблицей истинности. Там же показано окно задания параметров этого блока, в котором и задана соответствующая таблица истинности. Входные сигналы от блоков констант заданы как сигналы логического типа Boolean.

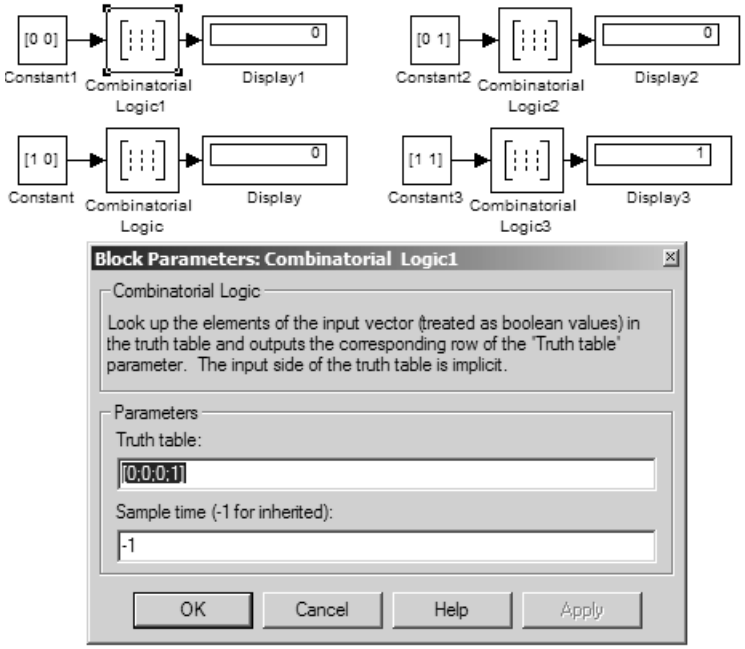


Рис. 6.7. Пример применения блока ***Combinatorial Logic***

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).

6.1.7. Блоки масштабирования Gain и Slider Gain

Для масштабирования данных (умножения их на заданный коэффициент передачи блока – константу) служат блоки **Gain** и **Slider Gain** (рис. 6.8). Блоки имитируют работу идеального усилителя. В блоке **Gain** константа вводится в окне параметров (по умолчанию 1), а в блоке **Slider Gain** ее можно выбирать с помощью ползунка, перемещаемого с помощью мыши. Для этого достаточно установить курсор мыши на ползунок, затем нажать левую клавишу мыши и, удерживая ее, начать перемещать ползунок влево или вправо. Множитель будет, соответственно, уменьшаться или возрастать. Для масштабирования матричных данных служит блок **Matrix Gain**. Пример применения этого блока также дан на рис. 6.8.

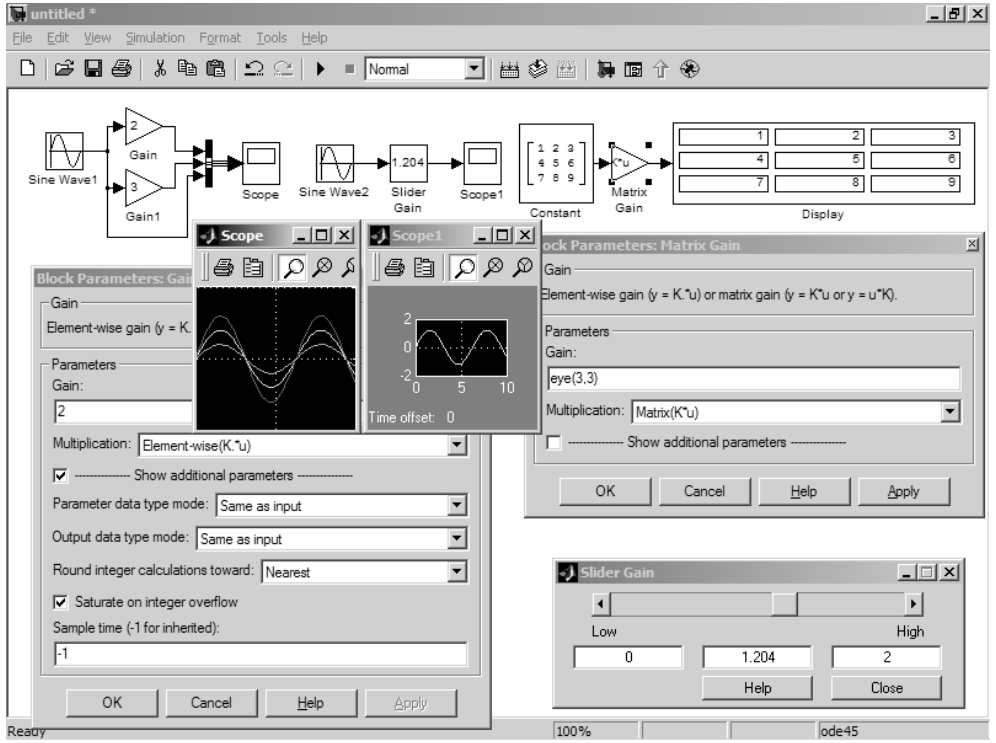


Рис. 6.8. Примеры применения блоков масштабирования

В окнах установки параметров блоков масштабирования можно задать коэффициент передачи Gain и способ выполнения операции умножения Multiplications:

- `Element-wise K*u` – поэлементное умножение;

- `Matrix K*u` – матричное умножение левосторонним оператором;
- `Matrix u*K` – матричное умножение правосторонним оператором;
- `Matrix K*u` – векторное умножение левосторонним оператором.

Окно можно расширить для ввода дополнительных параметров с помощью опции **Show additional parameters**. Эти параметры уже описывались и видны в расширенном окне блока **Gain**, показанном на рис. 6.8 в левом нижнем углу.

6.1.8. 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. 6.9.

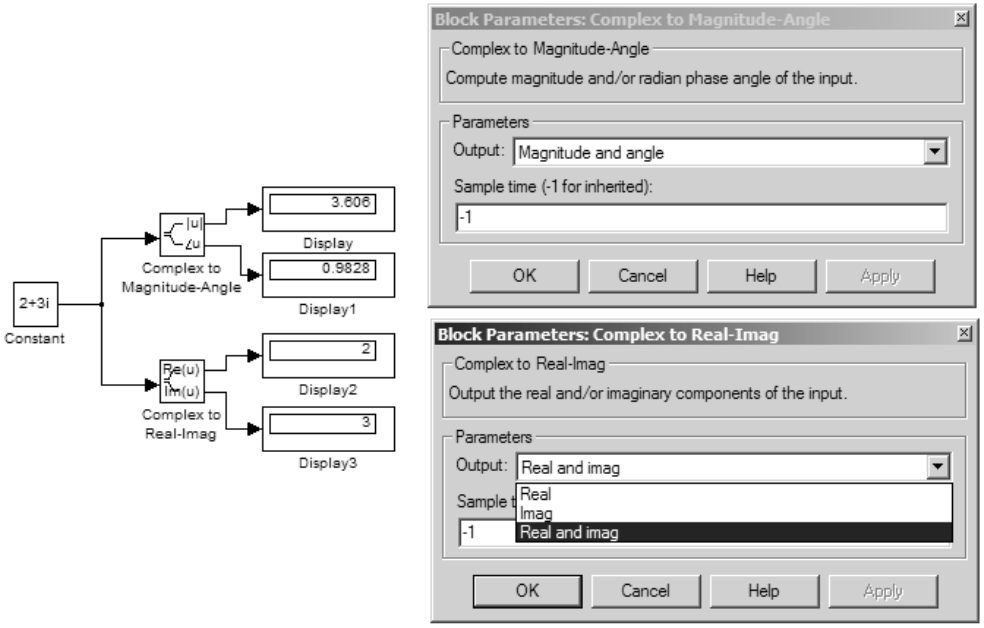


Рис. 6.9. Примеры применения блоков обработки комплексных данных

Можно вычислять либо любой из выходных параметров отдельно, либо оба одновременно. Эти блоки обычно используются для вычисления амплитудно-частотных (АЧХ) и фазо-частотных (ФЧХ) характеристик линейных систем.

6.1.9. Блоки Real-Image to Complex и Magnitude-Phase to Complex

Для задания комплексных чисел служат блоки **Real-Image to Complex** и **Magnitude-Phase to Complex**. Первый из них позволяет задать комплексное число по заданным действительной и мнимой частям. Примеры применения этого блока представлены на рис. 6.10.

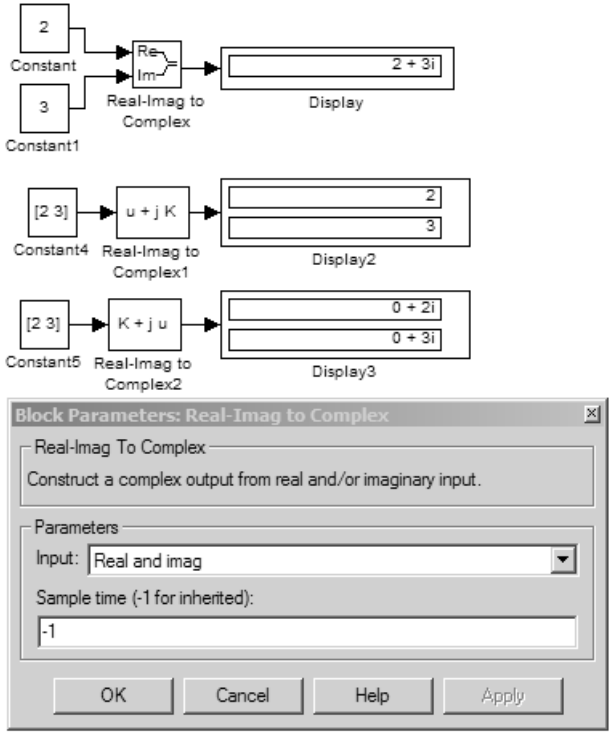


*Рис. 6.10. Примеры применения блока **Real-Image to Complex***

В окне задания параметров можно задать преобразование в комплексное число одновременно действительной или только мнимой составляющей комплексного числа. В последнем случае недостающий параметр задается в окне параметров блока. Пример применения этого блока показан на рис. 6.11.

Второй блок **Magnitude-Phase to Complex** служит для задания комплексного числа по заданным магнитуде и фазе. В окне задания параметров можно задать преобразование в комплексное число одновременно магнитуды и фазы, только магнитуды и только фазы. В последнем случае недостающий параметр задается в окне параметров блока.

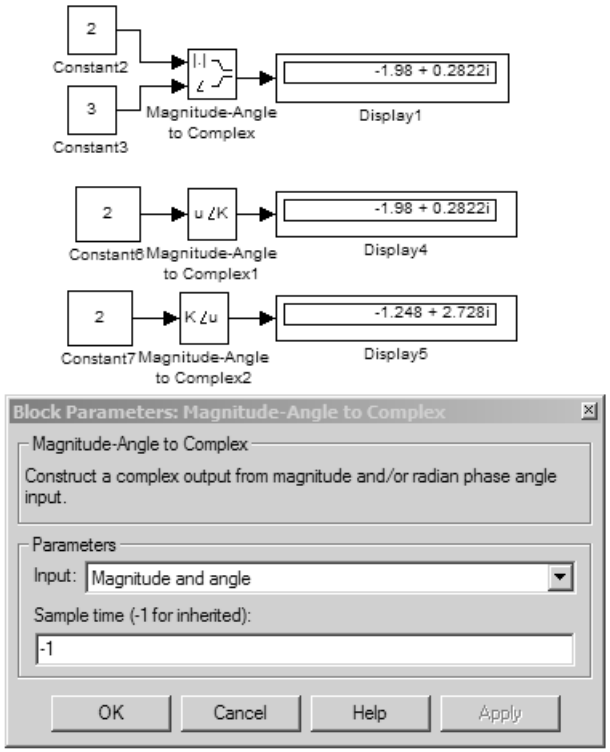


Рис. 6.11. Примеры применения блока ***Magnitude-Phase to Complex***

6.1.10. Блок поиска минимума и максимума MinMax

Для поиска в данных минимального и максимального значений служит блок **MinMax** (рис. 6.12).

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

6.1.11. 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. 6.13).

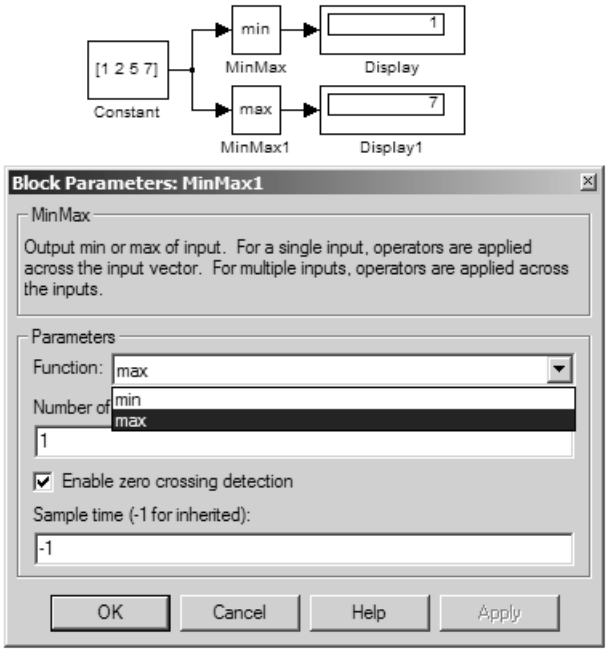


Рис. 6.12. Примеры применения блоков масштабирования

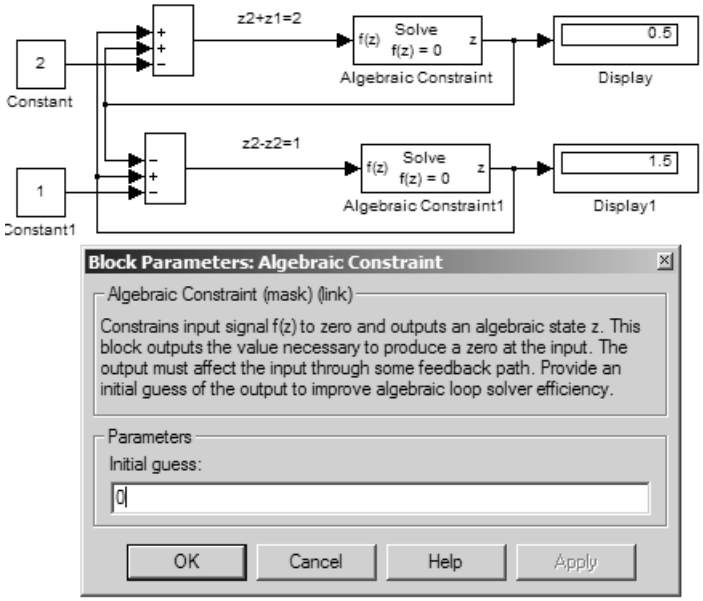


Рис. 6.13. Пример решения системы из двух уравнений

Модель рис. 6.13 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.

6.1.12. Матричные блоки Assignment, Matrix Concatenation и Reshape

Матричный блок `Assignment` служит для присваивания элементам векторов и матриц новых значений. Список параметров блока зависит от входных сигналов. На рис. 6.14 слева показано, как блок в векторе [1 2 3] меняет первое и второе значения на 5 и 6, выбирая их из второго вектора. Список индексов [1 2] задается в окне параметров блока `Assignment`.

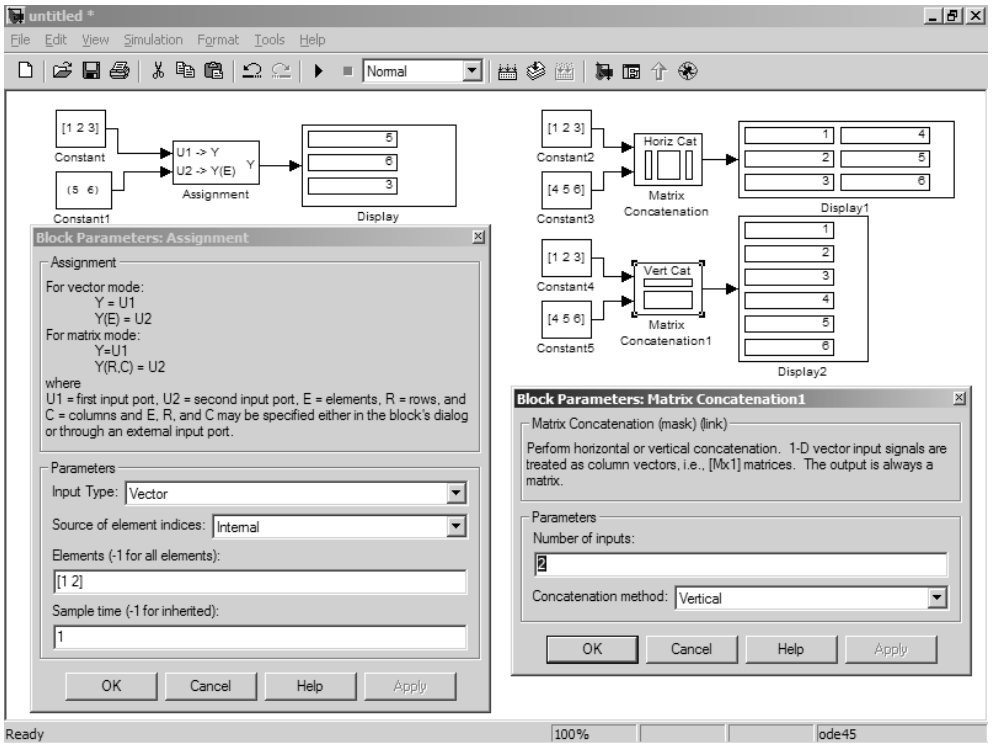


*Рис. 6.14. Примеры применения блоков **Assignment** и **Matrix Concatenation***

В окне параметров задается также то, откуда берутся данные индексации – изнутри блока **Internal** и извне **External**. Во втором случае у блока формируется дополнительный вход для задания индексов. Возможен также параметр **Elements** – список индексов входного вектора, передаваемых на выход блока (для передачи всех элементов задается –1). Возможна также индексация с помощью матрицы – заинтересованный в этом читатель легко разберется с ней.

Блок **Matrix Concatenation** служит для объединения (конкатенации) векторов и матриц. Возможно объединение по горизонтали и по вертикали. Простой пример объединения двух векторов представлен на рис. 6.14 (справа). В окне параметров этого блока указываются число входов и тип объединения.

Блок изменения размерности **Reshape** позволяет изменять размерность вектора или матрицы. Несколько примеров применения этого блока показаны на рис. 6.15. В первых трех примерах матрица превращается в одномерный массив, вектор-строку и вектор-столбец. В последнем примере вектор из шести элементов превращается в матрицу, содержащую три строки и два столбца. Виды преобразований задаются в окне параметров блока путем выбора из списка.

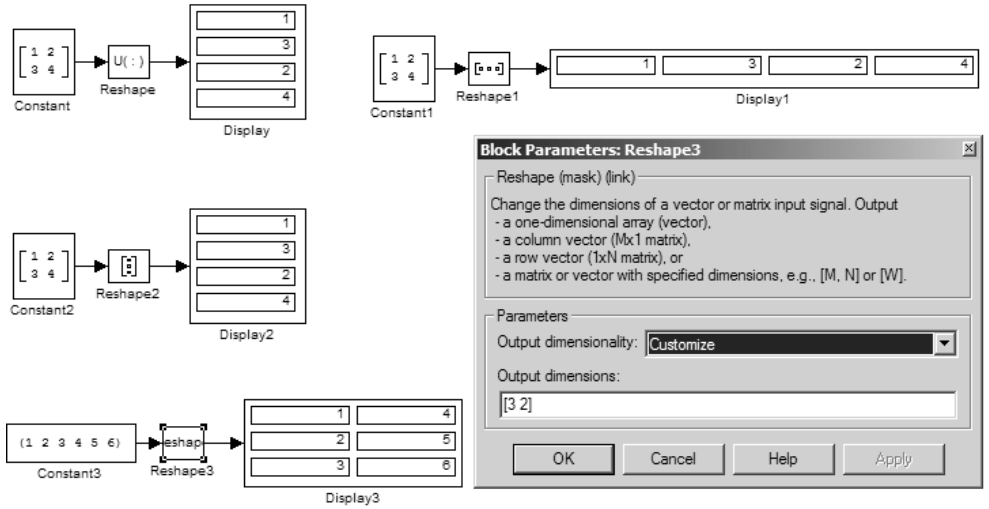


*Рис. 6.15. Примеры применения блока **Reshape***

6.1.13. Блок вычисления значений полинома Polynomial

Блок `Polynomial` служит для вычисления значений полинома по заданному вектору его коэффициентов – рис. 6.16. Например, для задания полинома $x^2 + 2x + 3$ надо задать вектор его коэффициентов в виде [1 2 3]. Именно этот вектор и задан в качестве единственного параметра окна параметров блока `Polynomial`.

На рис. 6.16 показано вычисление значения заданного полинома и построение его графика с помощью графопостроителя.

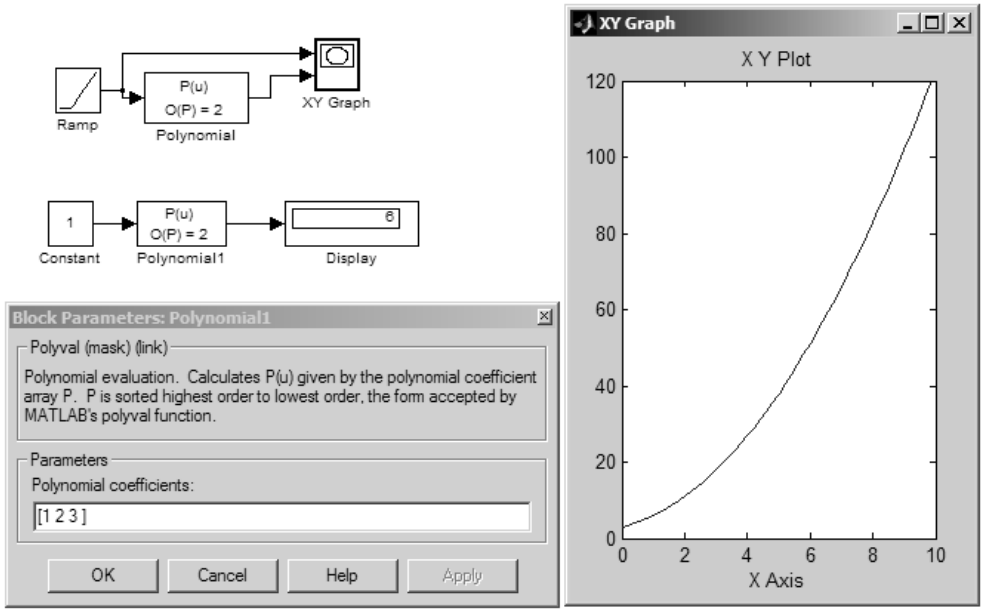


Рис. 6.16. Примеры применения блока Polynomial

6.2. Библиотека непрерывных блоков Continuous

6.2.1. Раздел библиотеки Continuous

Непрерывные (Continuous) блоки 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. 6.17 представлен раздел библиотеки **Continuous** с непрерывными компонентами.

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.

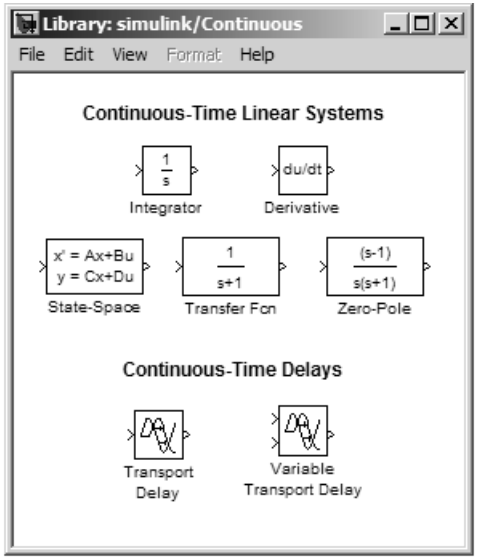


Рис. 6.17. Раздел библиотеки с непрерывными блоками

6.2.2. Дифференцирующий блок Derivative

Дифференцирующий блок **Derivative** служит для численного дифференцирования входных данных (сигналов). На рис. 6.18 дан пример последовательного интегрирования и дифференцирования пилообразных импульсов, а также окно установки параметров дифференцирующего блока.

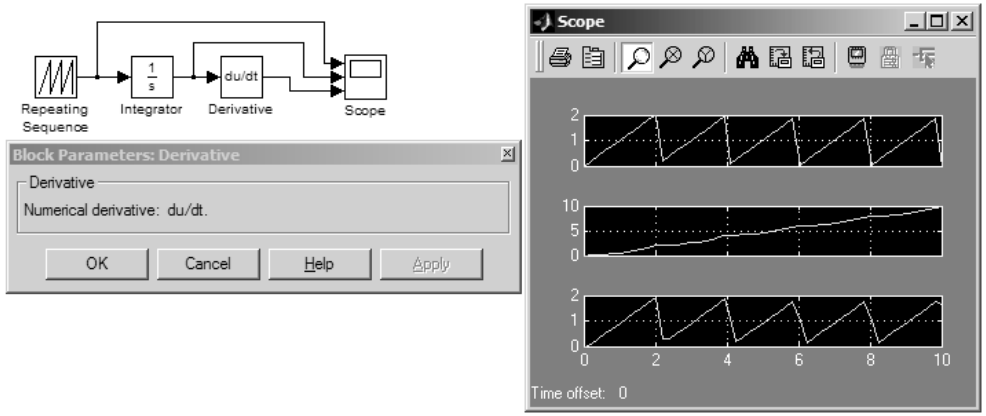


Рис. 6.18. Интегрирование пилообразных импульсов с последующим дифференцированием

Дифференцирование выполняется приближенно по простой формуле Эйлера:

$$\frac{du}{dt} = \frac{\Delta u}{\Delta t},$$

где Δu – приращение входного сигнала за время дискретизации Δt.

Нетрудно заметить, что никаких параметров блок дифференцирования не имеет, и его окно установки параметров дает только информацию о назначении блока. Из осциллограмм рис. 6.18 можно заметить, что как интегрирование (см. ниже), так и дифференцирование выполняются неидеально, но тем не менее форма пилообразного сигнала после последовательного выполнения этих операций практически восстанавливается.

6.2.3. Интегрирующий блок Integrator

Блок интегрирования **Integrator** выполняет функции интегрирования входных данных (сигналов). На рис. 6.19 приведен еще один пример совместного применения блоков интегрирования и дифференцирования – для входного сигнала в виде прямоугольных импульсов. На рис. 6.19 показано также окно установки параметров интегрирующего блока.

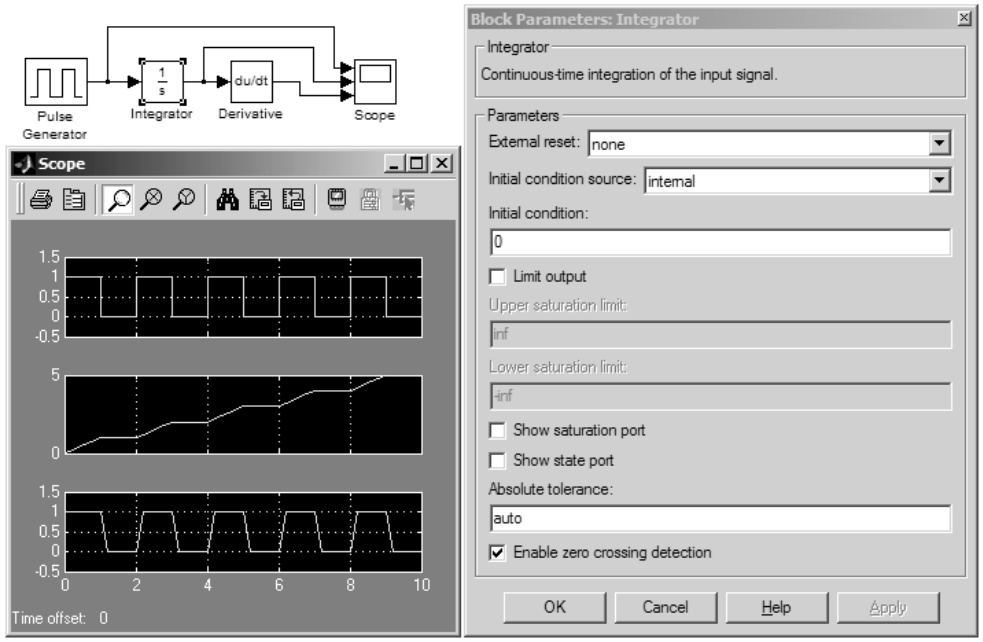


Рис. 6.19. Интегрирование прямоугольных импульсов с последующим дифференцированием

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 $+\infty$);
- **Lower saturation limit** – нижний предел ограничения выходного сигнала (по умолчанию **-inf**, то есть $-\infty$);
- **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).
- **Enable zero crossing detections** – включение проверки переходов через нуль.

Большое число параметров у блока интегратора обеспечивает возможность его удобной настройки для моделирования процессов интегрирования.

6.2.4. Блок задания линеаризованной модели State-Space

В ходе моделирования Simulink автоматически составляет некоторую исходную модель системы, которая представляет собой систему нелинейных алгебраических уравнений. Для вычисления изменения на каждом малом шаге эта система линеаризируется и приводится к матричной системе уравнений:

$$d\boldsymbol{x}/dt = \boldsymbol{A}\cdot\boldsymbol{x} + \boldsymbol{B}\cdot\boldsymbol{u}, \quad (6.1)$$

$$\boldsymbol{y} = \boldsymbol{C}\cdot\boldsymbol{x} + \boldsymbol{D}\cdot\boldsymbol{u}, \quad (6.2)$$

где $\boldsymbol{x}$ – вектор состояния, $\boldsymbol{u}$ – вектор входных воздействий и $\boldsymbol{y}$ – 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. 6.20. Здесь m – число входов, n – число со-

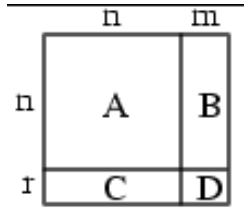


Рис. 6.20. Структура матриц линеаризованной матричной системы уравнений

стояний системы, r – число выходов. Рисунок 6.20 дает представление о размерах матриц в линеаризованной матричной системе уравнений.

На рис. 6.21 показан пример применения блока **State-Space**. Блок задает линейную систему второго порядка, матрицы и векторы в описании пространства состояний указаны в окне параметров этого блока. Основными параметрами здесь являются матричные коэффициенты, по умолчанию равные 1, и параметр **Initial condition**, задающий вектор начального состояния (по умолчанию 0).

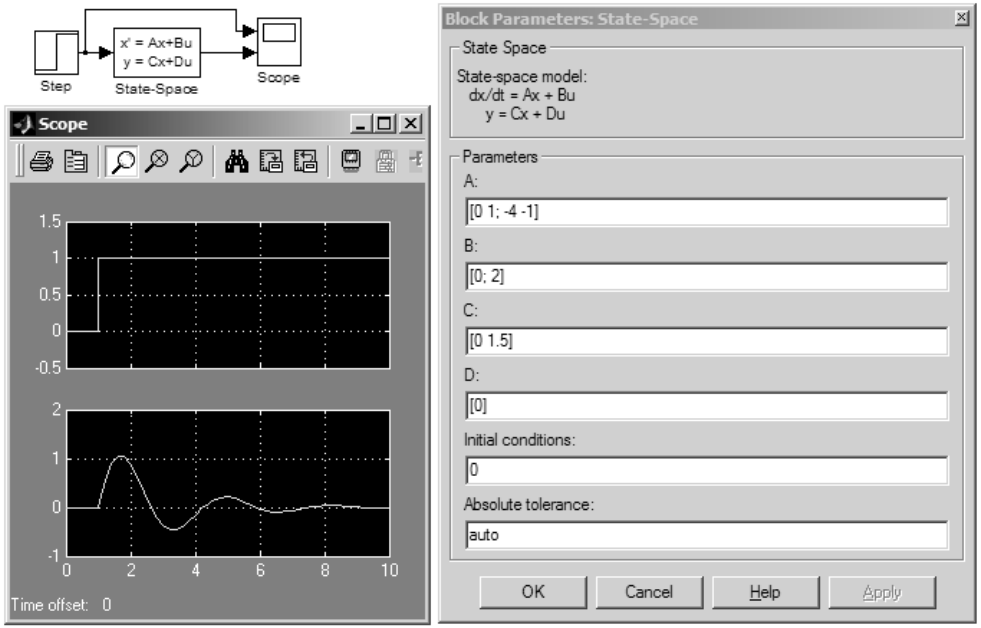


*Рис. 6.21. Пример применения блока **State-Space***

6.2.5. 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. На рис. 6.22 показан пример на применение блока передаточной функции. Вид блока показан после задания его параметров в окне параметров. Осциллограф показывает воздействие в виде ступеньки и передаточную характеристику звена, заданного передаточной функцией.

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

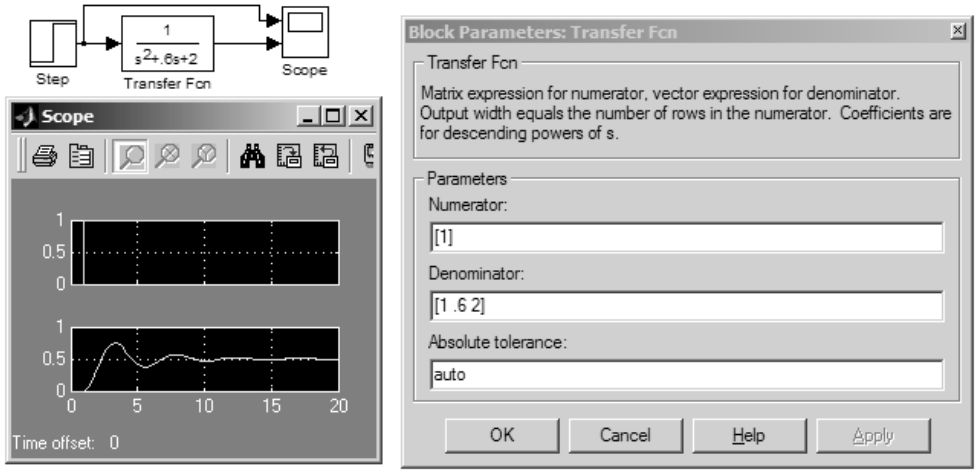
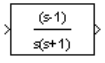


*Рис. 6.22. Пример применения блока **Transfer Fcn***

Если коэффициенты числителя полинома заданы вектором, то выходной сигнал, как и входной, будет скалярным. В случае задания коэффициентов числителя матрица блок задает семейство передаточных характеристик с одинаковыми полиномами знаменателя. В этом случае выходной сигнал является векторным, и число строк матрицы числителя задает размерность выходного сигнала.

6.2.6. Блок Zero-Pole

Блок **Zero-Pole** служит для создания передаточной функции с заданными полюсами и нулями. Исходный вид блока и его передаточная характеристика в общем виде представлены на рис. 6.23. После задания конкретной передаточной функции вид блока меняется.



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

*Рис. 6.23. Блок **Zero-Pole** и его передаточная функция*

Пример применения блока и окно его параметров показаны на рис. 6.24. В этом окне задаются списки нулей (поле **Zeros**) и полюсов (поле **Poles**) передаточной характеристики, а также коэффициент передачи (поле **Gain**).

Нули и полюса передаточной функции в блоке могут задаваться парами комплексно-сопряженных чисел. Нули могут быть заданы матрицами. Начальные условия при применении данного блока обычно полагаются нулевыми.

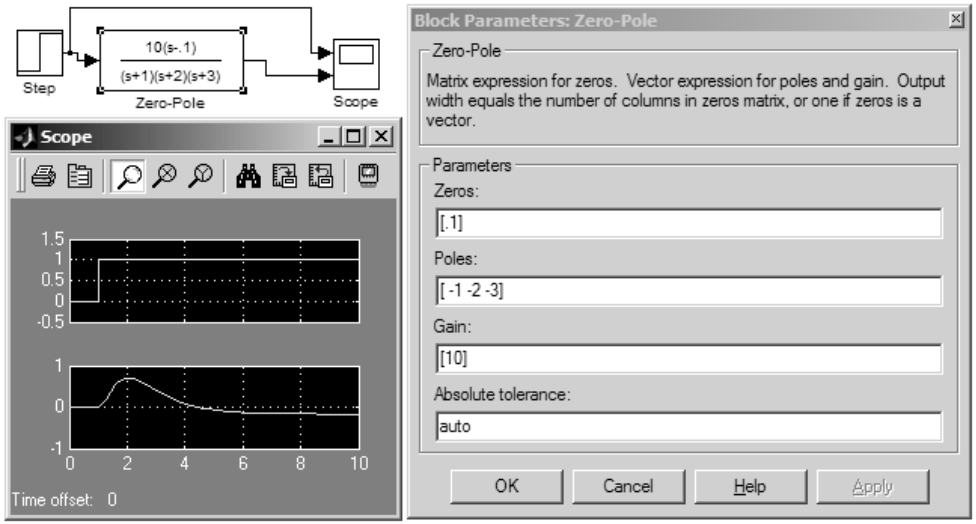


Рис. 6.24. Пример применения блока ***Zero-Pole***

6.2.7. Блок фиксированной задержки Transport Delay

Блок фиксированной задержки **Transport Delay** обеспечивает временную задержку входного сигнала на заданное время. Пример применения блока и окно установки его параметров показаны на рис. 6.25.

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 байта);
- **Pade order (for linearization)** – порядок линеаризации Паде (по умолчанию 0, но может задаваться как целое положительное *n* для повышения точности линеаризации).

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

6.2.8. 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. 6.26).

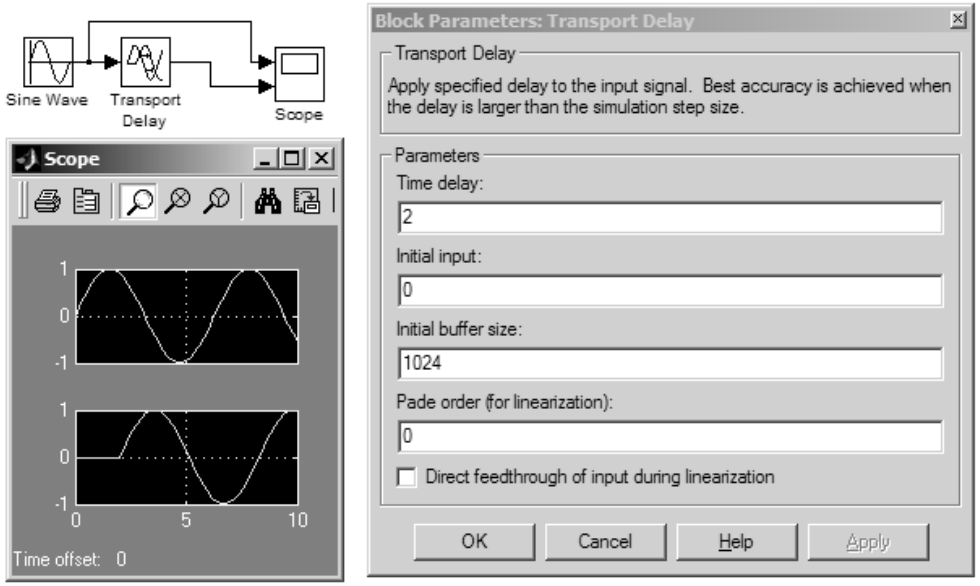


Рис. 6.25. Применение блока ***Transport Delay***

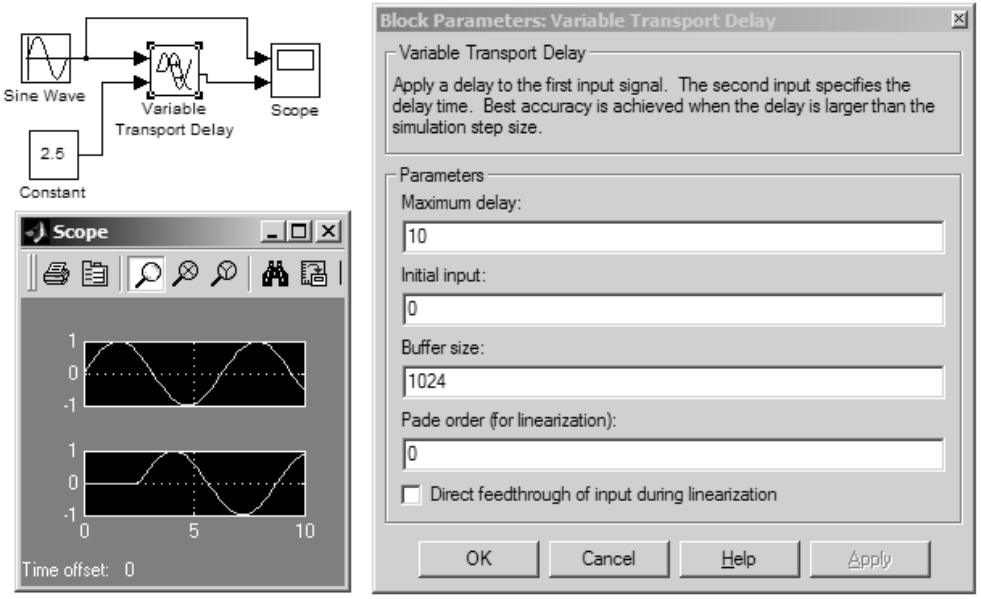


Рис. 6.26. Применение блока ***Variable Transport Delay***

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. 6.26 демонстрирует задержку на 2,5 такта эталонной частоты, что задается константой 2,5 на входе управления блока управляемой задержки.

6.3. Блоки задания таблиц

6.3.1. Обзор блоков таблиц

Раздел библиотеки **Look-Up Tables** содержит компоненты таблиц. Следует отметить, что в Simulink 5/6 этот раздел заметно сокращен – из него изъяты блоки задания функций и блок сменил название. В результате число блоков раздела сократилось до 6. Блоки этого раздела показаны в окне рис. 6.27.

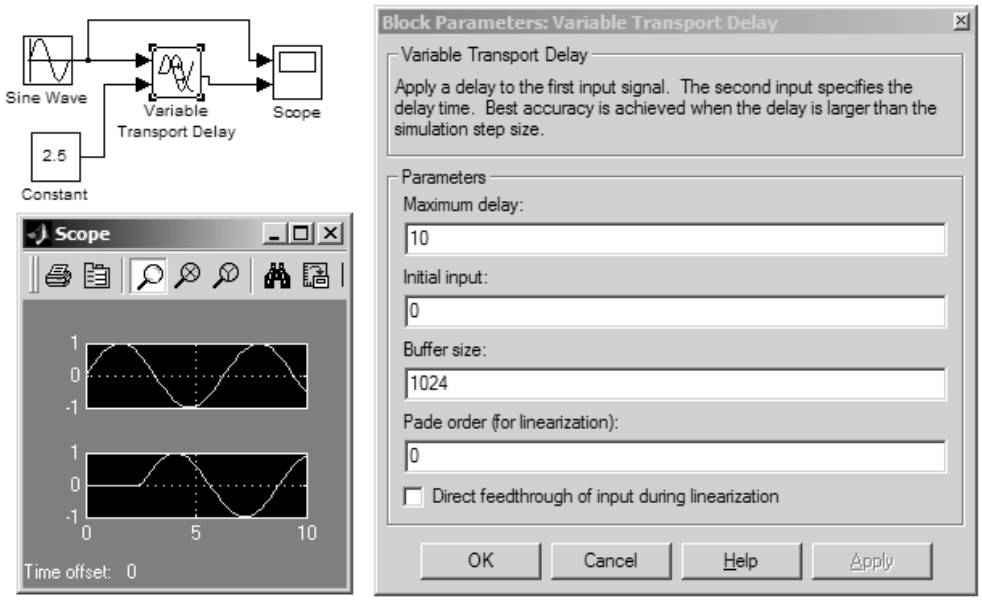


Рис. 6.27. Блоки таблиц

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.

6.3.2. 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. 6.28 дан пример определения функции гиперболического тангенса для значений входного аргумента от –5 до 5 с шагом 1. Амплитуда синусоидального сигнала задана равной 2 В.

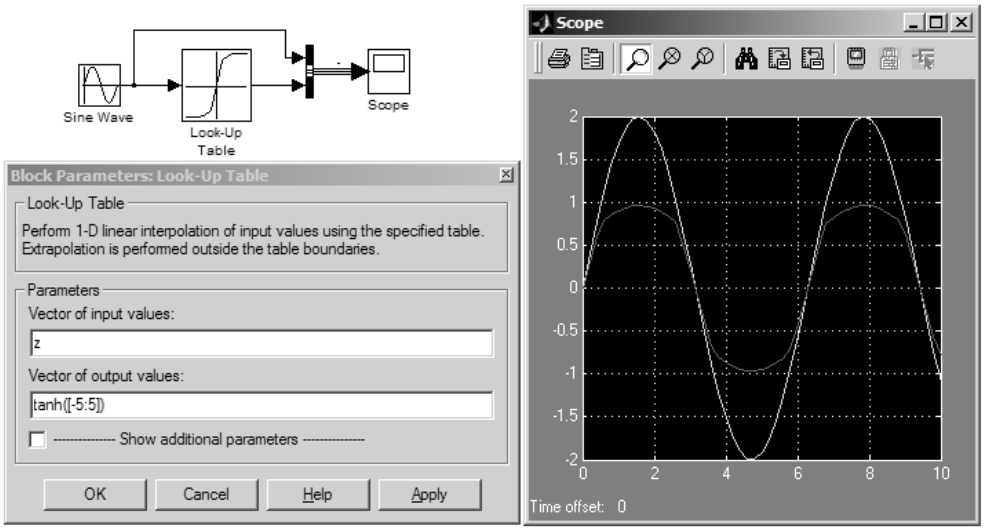


Рис. 6.28. Пример применения блока задания одномерной таблицы

В окне параметров блока (рис. 6.28 снизу) задаются векторы входного и выходного сигналов. Они могут задаваться дискретными значениями (например, [1 3 8 10]), в виде диапазона (например, [–5,5]), выражения (например, tanh([–5:5])) и т. д. Эти векторы задают характер зависимости выходного сигнала от входного.

При необходимости можно, установив птичку в окне опции **Show additional parameters**, расширить окно и получить доступ к дополнительным параметрам – рис. 6.29.

В расширенном окне параметров можно установить ряд опций:

- `Look-up method` – метод расчета выходных значений. Он выбирается из списка:
  - `Interpolation-Extrapolation` – линейная интерполяция и экстраполяция;

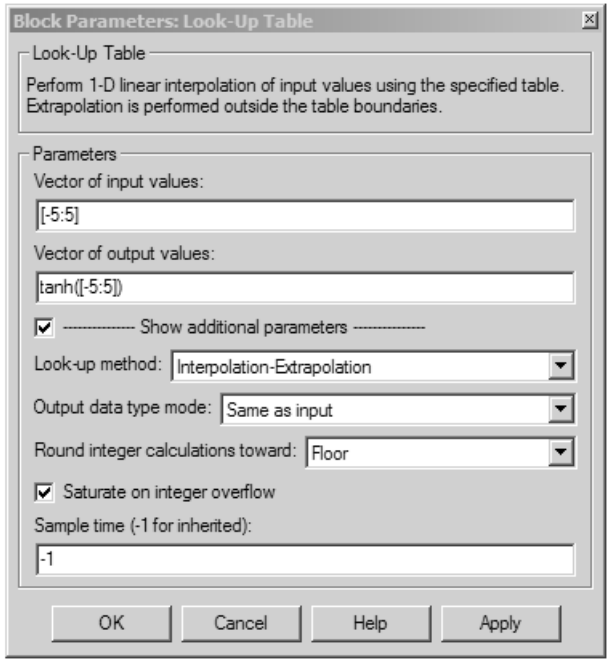


Рис. 6.29. Окно параметров блока задания одномерной таблицы

  - `Interpolation-Use End Values` – линейная интерполяция внутри заданного интервала входных значений и ограничение выходного сигнала по конечным значениям за пределами заданного интервала;
  - `Use Input Nearest` – отсутствие интерполяции, при вычислении выходного сигнала берется ближайшее табличное значение;
  - `Use Input Bellow` – отсутствие интерполяции, при вычислении выходного сигнала берется ближайшее меньшее табличное значение;
  - `Input Above` – отсутствие интерполяции, при вычислении выходного сигнала берется ближайшее большее табличное значение.
- `Output data mode` – выбор типа выходных данных из списка:
  - `inherited via back propagation` – выходной сигнал наследуется от блока назначения;
  - `same as input` – выходной сигнал наследуется от блока, подключенного ко входу;
  - `specify via dialog` – со спецификацией в диалоге;
  - — выходной сигнал берется стандартного типа, например int8 и т. д.
- `Output data type` – выбор типа выходных данных из списка.
- `Output scaling value` – задание масштабирующей величины со смещением, которое допустимо не указывать.
- `Lock output scaling against changes by autoscaling tool` – блокировка масштабирования.

- `Rounding integer calculations toward` – выбор метода округления из списка (`Zero` – округление отсутствует, `Nearest` – до ближайшего целого, `Floor` – до ближайшего меньшего целого и `Ceiling` – до ближайшего большего целого).
- `Saturate on integer overflow` – ограничение переполнения целых чисел.

6.3.3. 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, предусмотрены линейная интерполяция и экстраполяция входных данных. Рисунок 6.30 иллюстрирует применение этого блока.

В окне параметров блока двумерной таблицы задаются векторы индексов строк **Row** и столбцов **Column**, а также матрица данных таблицы **Table**. Возможно также расширение окна для ввода дополнительных параметров, аналогичных описанным выше.

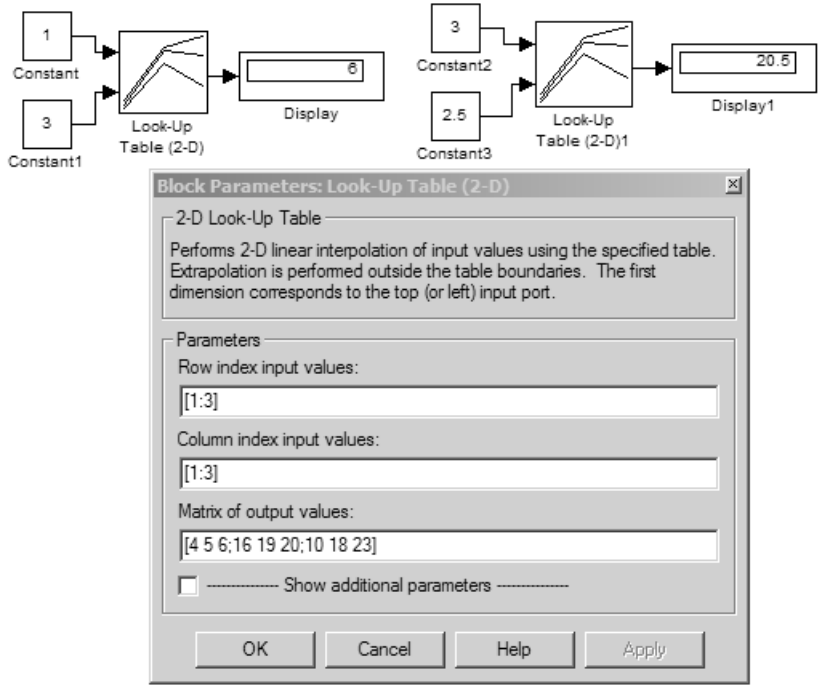


Рис. 6.30. Блок задания двумерной таблицы

6.3.4. 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, показанными на рис. 6.31.

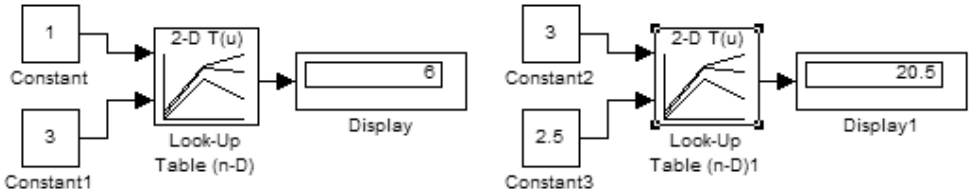


Рис. 6.31. Применение блока задания многомерных таблиц

Окно параметров этого блока показано на рис. 6.32. В связи с расширенными возможностями этого блока число его параметров достаточно велико в сравнении с ранее рассмотренными блоками.

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

- **Number of table dimension** – размерность таблицы;
- **First input (row) breakpoint set** – первые установки входа (индексы строки);
- **Second (column) input breakpoint set** – вторые установки входа (индексы столбца);
- **Index search method** – метод просмотра индекса (Evenly Spaced Points – обзор по равномерно распределенным точкам, Linear Search – линейный обзор, Binary Search – бинарный обзор);
- **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 интерполяции, `Linear` – линейная интерполяция, `Cubic spline` – кубическая сплайновая интерполяция);
- **Action for out of range input** – действие при выходе входных данных за допустимые пределы (выбирается из раскрывающегося списка: `None` – без сообщения, `Warning` – предупреждение, `Error` – сообщение об ошибке, по умолчанию – `Warning`).

Кроме того, в окне параметров имеются два флажка:

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

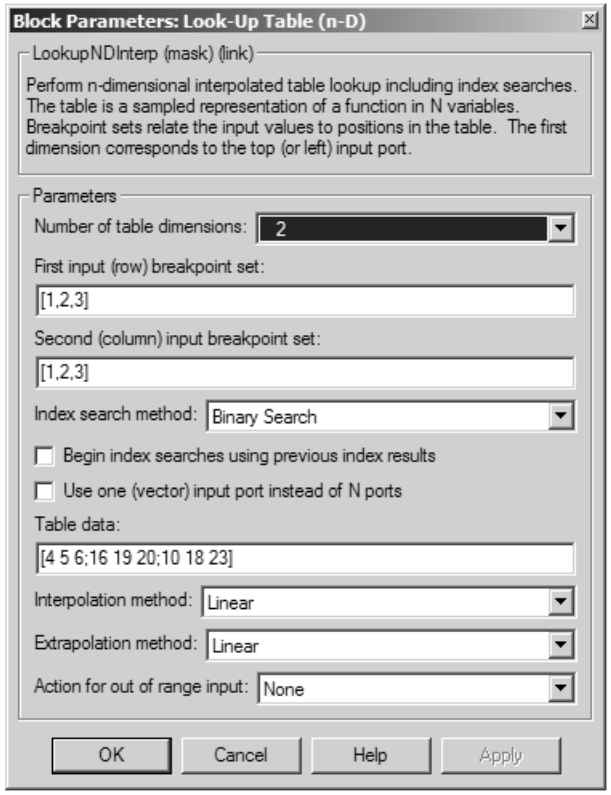


Рис. 6.32. Окно параметров блока задания многомерных таблиц

- **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.

6.3.5. 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. 6.33.

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

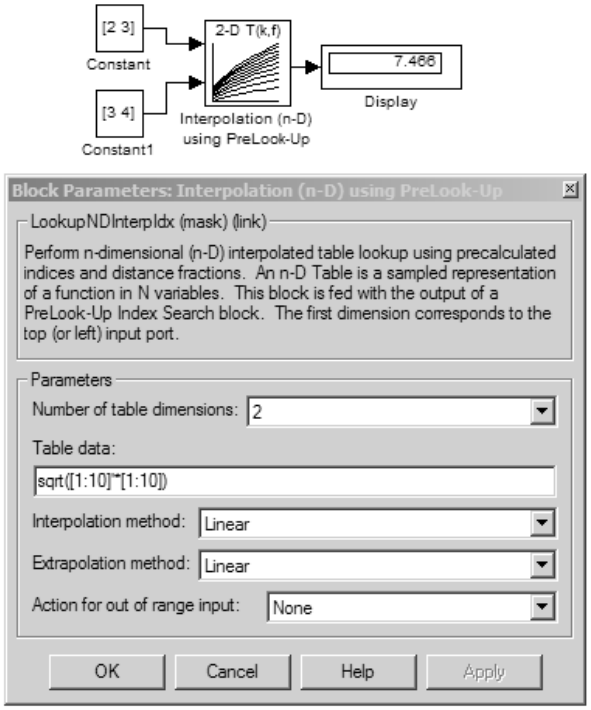


Рис. 6.33. Применение блока ***Interpolation (n-D) using PreLoop-Up***

6.3.6. 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. 6.34.

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 размера 22);
- **Make table an input** (опция) – сделать таблицу входом;
- **Table data** – массив данных таблицы;
- **Action for out of range input** – действие при выходе входных данных за допустимые пределы (выбирается из раскрывающегося списка: `None` – без сообщения, `Warning` – предупреждение, `Error` – сообщение об ошибке, по умолчанию – `Warning`).

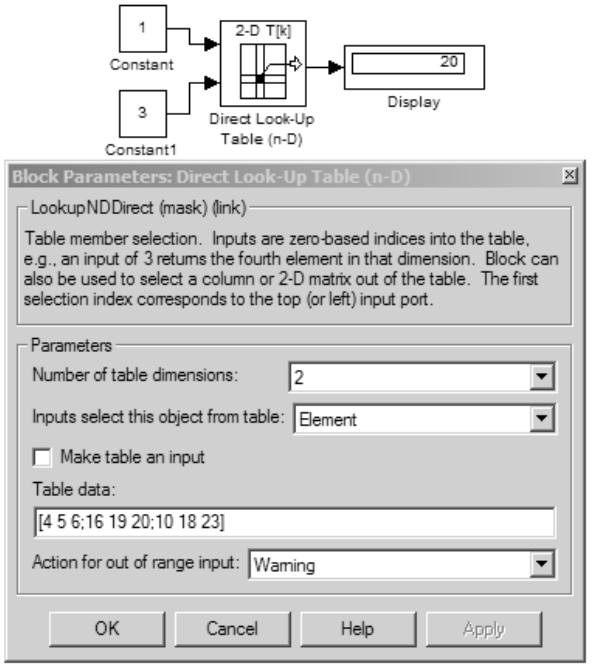


Рис. 6.34. Применение блока ***Direct Loop-Up Table (n-D)***

6.3.7. 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. 6.35, где показана обработка данных вектора из трех констант [12, 45, 78] (не спутайте этот вектор с ссылками на литературу).

Для этого блока задаются следующие параметры:

- **Breakpoint data** – вектор (или ранжированная переменная) для узловых точек;
- **Index search method** – метод поиска индексов (`Binary Search` – бинарный обзор, `Linear Search` – линейный обзор или `Evently Spaced Points` – обзор 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);

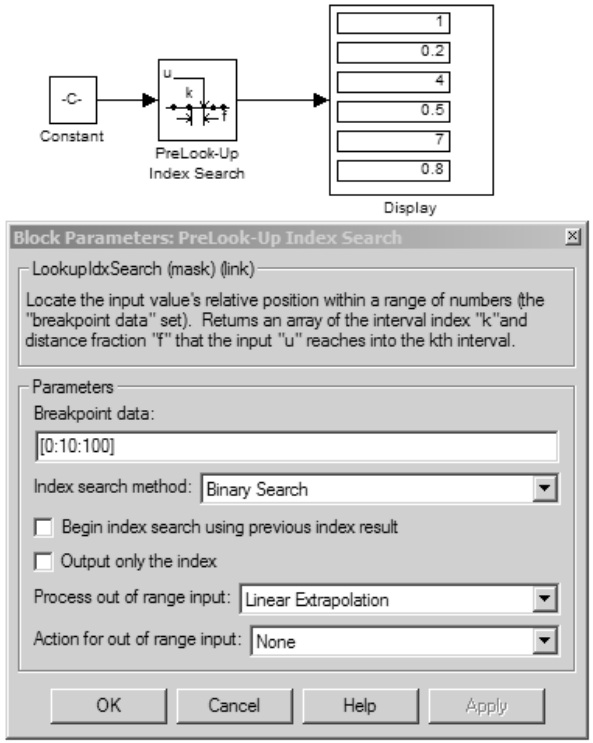


Рис. 6.35. Применение блока ***PreLook-Up Index Search***

- **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.

6.4. Блоки задания функций пользователя

6.4.1. Обзор функций пользователя

Новый раздел библиотеки в Simulink 5/6 – раздел по функциям пользователя – **User Defined Functions** (рис. 6.36). Новым этот раздел можно назвать условно, поскольку в него просто помещены четыре блока для задания функций пользователя, которые ранее были в других разделах библиотеки блоков.

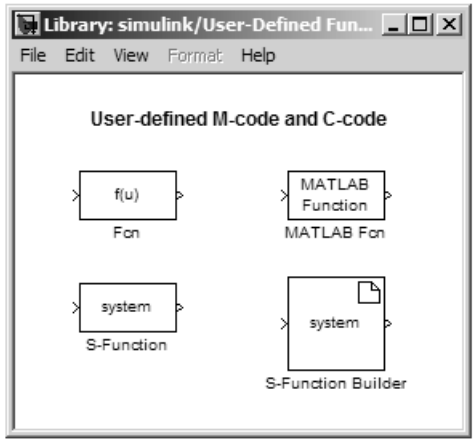


Рис. 6.36. Применение блока ***PreLook-Up Index Search***

6.4.2. Блок задания функции Fcn

Блок **Fcn** служит для задания функций одной переменной u или ряда переменных u(i). На рис. 6.37 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, приня-

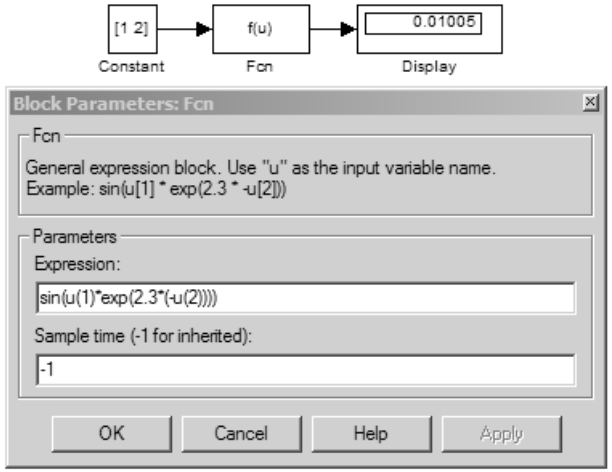


Рис. 6.37. Блок задания функции ***Fcn***

тым для описания функций на языке C. Допустимые операторы перечислены ниже 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**.

6.4.3. Блок задания функции MATLAB Fcn

Блок задания функции **MATLAB Fcn** служит для задания функций одной переменной u или ряда переменных u(i) 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 6.38 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. 6.38) содержит описание правил задания функции и раздел параметров **Parameters**. В разделе **MATLAB function** задаются 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. 6.38), 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**).

6.4.4. Блок задания S-функций

Модели расширения Simulink в конечном итоге являются программами, называемыми *S-функциями*. Блок задания S-функций – одновременно самый простой и самый сложный блок. Простой, потому что окно его параметров, представленное на рис. 6.39, содержит лишь поля для ввода наименования S-функции и ее параметров.

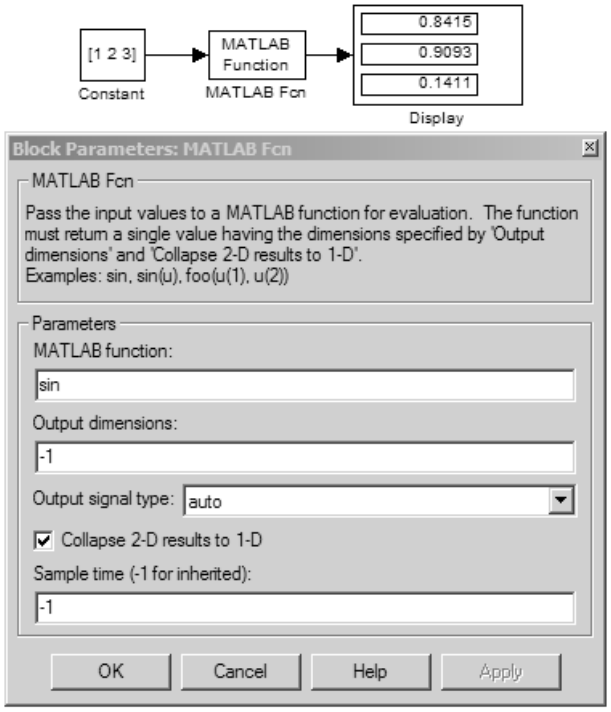


Рис. 6.38. Блок задания функции MATLAB Fcn

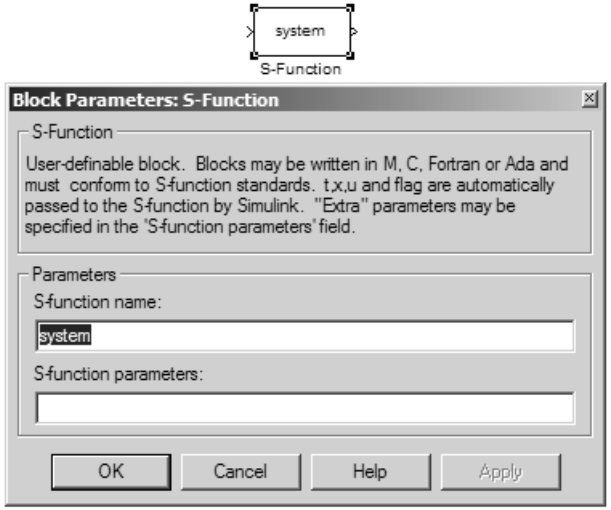


Рис. 6.39. Блок S-функций и окно его параметров

Сложность S-функций обусловлена тем, что они являются вполне самостоятельными программами, написанными на языках MATLAB, C, Ада, Фортран и др., 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 нелегко представить ситуацию, когда задачу нельзя решить с помощью стандартных библиотечных блоков. А потому возможности S-функций интересны лишь достаточно квалифицированным пользователям, владеющим навыками программирования.

6.4.5. Примеры применения S-функций

Для получения доступа к демонстрационным примерам на применение S-функций достаточно в разделе справки **Demos** найти и исполнить команду **S-function examples**. В Simulink 4 эта команда выводила окно с весьма внушительным набором демонстрационных примеров. В Simulink 5 примеры сгруппированны по реализациям на разных языках программирования, так что окно примеров выглядит довольно просто – рис. 6.40.

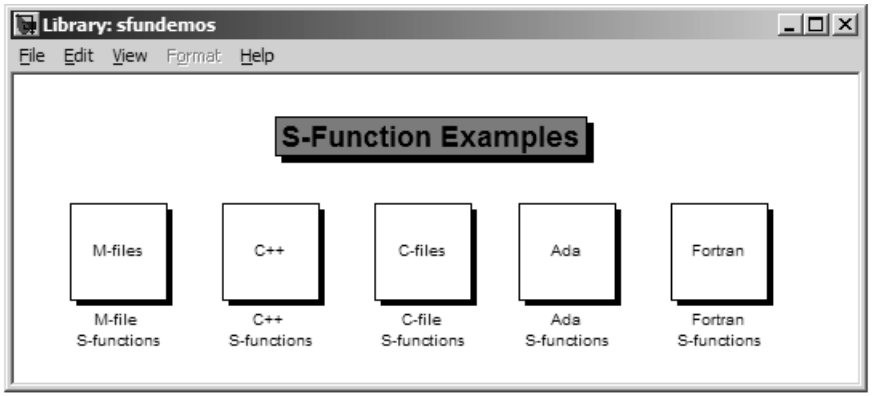


Рис. 6.40. Основное окно примеров применения S-функций

Полное описание S-функций и их демонстрационных примеров явно выходит за рамки тематики и объема этой книги, посвященной визуально-ориентированному математическому моделированию с помощью пакета расширения Simulink. Поэтому ограничимся демонстрацией лишь пары примеров, относящихся к применению S-функций, записанных на языке программирования системы MATLAB, детально описанном в этой книге. Выбрав на рис. 6.40 блок примеров с названием M-files, получим окно, представленное на рис. 6.41.

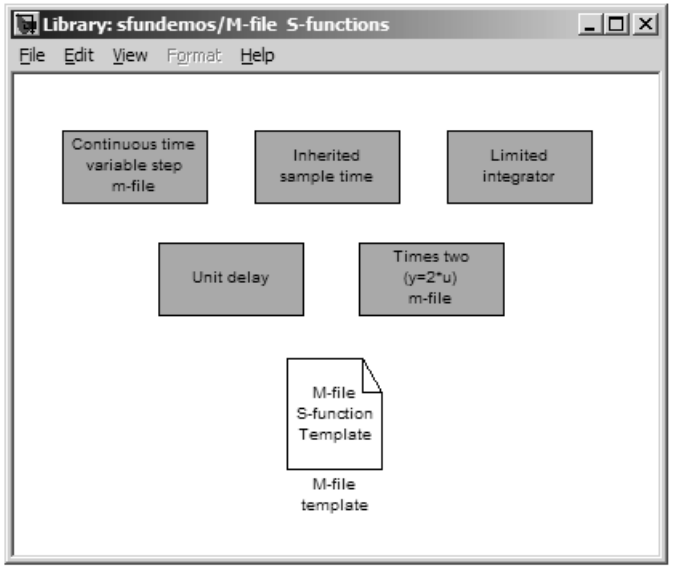


Рис. 6.41. Блок примеров применения S-функций, написанных в виде M-файла

Один из примеров этой группы представлен на рис. 6.42. Здесь дано задание элемента временной задержки на один шаг (такт) моделирования. Показано действие задержки на примере синусоидального сигнала. Задержка реализована M-файлом sfundsc2.m, листинг которого виден в окне редактора M-файлов в нижней части рисунка. Для вывода листинга этого файла (MATLAB-программы) достаточно активизировать надпись под моделью.

Другой пример (рис. 6.43) иллюстрирует реакцию системы, заданной (в виде S-функции) системой из двух дифференциальных уравнений. Представлены результат моделирования в виде осциллограммы и начало задания S-функции на языке MATLAB – функция задается в окне редактора M-файлов. Заинтересованный читатель может просмотреть этот файл и даже модифицировать его – система MATLAB является открытой системой, и любые M-файлы можно просматривать и модифицировать. В представленном примере вы можете просмотреть также пример задания S-функции на языке программирования C.

6.4.6. Блок создания S-функций S-Function Builder

Для создания S-функций, написанных на языке C, служит блок **S-Function Builder**. Этот блок и его окно показаны на рис. 6.44.

Окно блока имеет ряд вкладок, заполнение которых позволяет последовательно определить S-функцию, начиная с инициализации и заканчивая определением

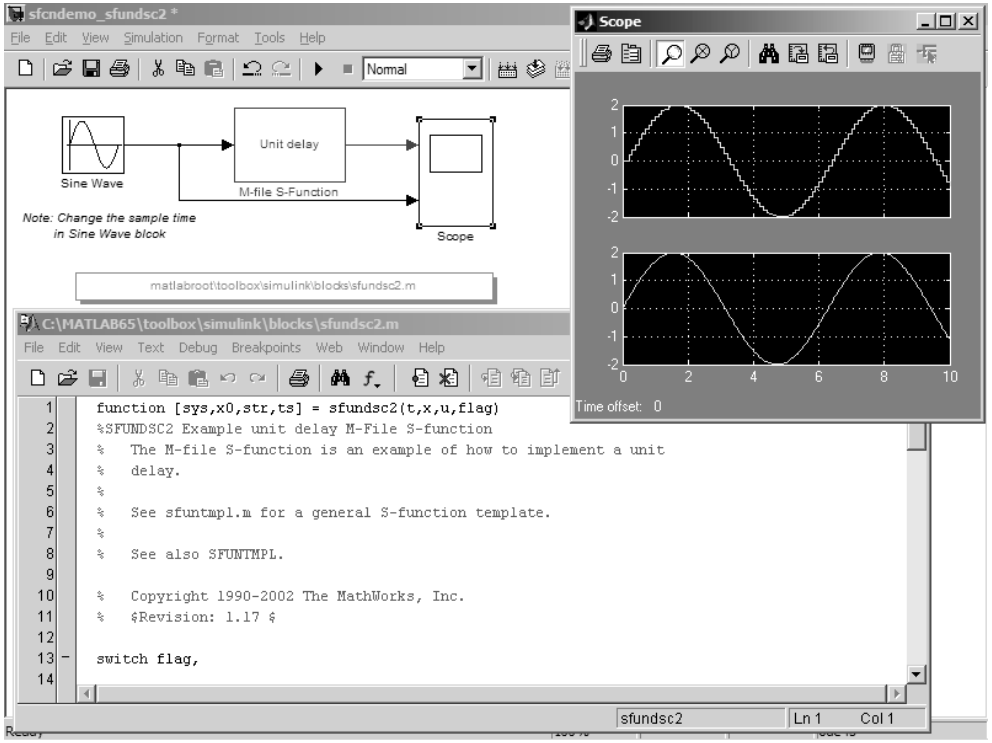


Рис. 6.42. Пример реализации блока S-функции, дающего задержку сигнала на один шаг

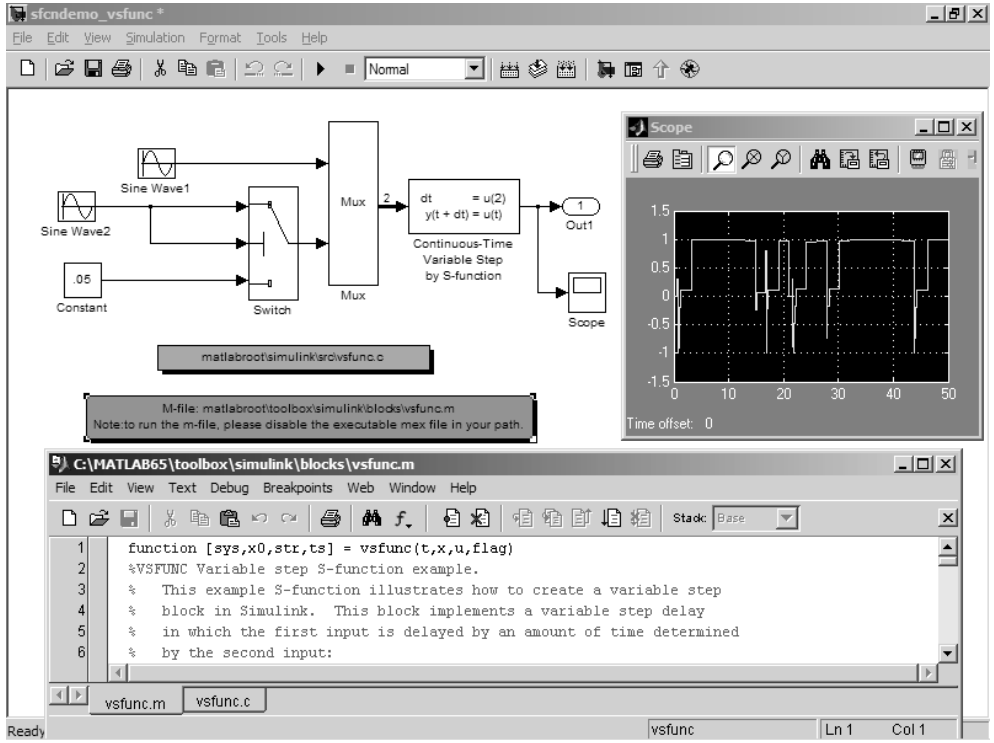


Рис. 6.43. Пример реализации блока S-функции, решающей систему из двух дифференциальных уравнений

библиотеки, куда заносится блок с S-функцией. Для компиляции D-функции достаточно нажать кнопку **Build (Создать)**. Однако для компиляции должны быть установлены специальные средства, рассмотрение которых выходит за рамки данной книги. Подготовка S-функций достаточно подробно описана в [33], так что заинтересованный читатель может продолжить знакомство с этой возможностью Simulink по данному источнику.

6.5. Новые блоки библиотеки Simulink 6/7

6.5.1. Расширенная библиотека математических блоков

Библиотека математических блоков в Simulink 6.* была дополнена рядом новых блоков. Окно этой библиотеки показано на рис. 6.45.

Большинство новых блоков имеют вполне очевидные функции и в особых пояснениях не нуждаются. Таковы блоки добавления **Add**, вычитания **Subtract**, суммирования элементов **Sum of Elements**, умножения элементов **Product of Elemtnts**, деления **Divide**, выделения знака **Sign**, вычисления абсолютного значения **Abs**, смены знака **Unary Minus** и вычисления синуса **Sine Wave Function**. Работу этих блоков читатель может легко проверить самостоятельно.

6.5.2. Блоки раздела Logic and Bit Operations

Пакет расширения Simulink 6 имеет три новых раздела библиотеки. Один из них Commonly Use Blocks, представленный на рис. 6.46, является просто сборкой наиболее часто применяемых блоков. А вот в раздел Logic and Bit Operations перенесены блоки логических и побитовых операций. При этом состав блоков заметно расширился – рис. 6.45.

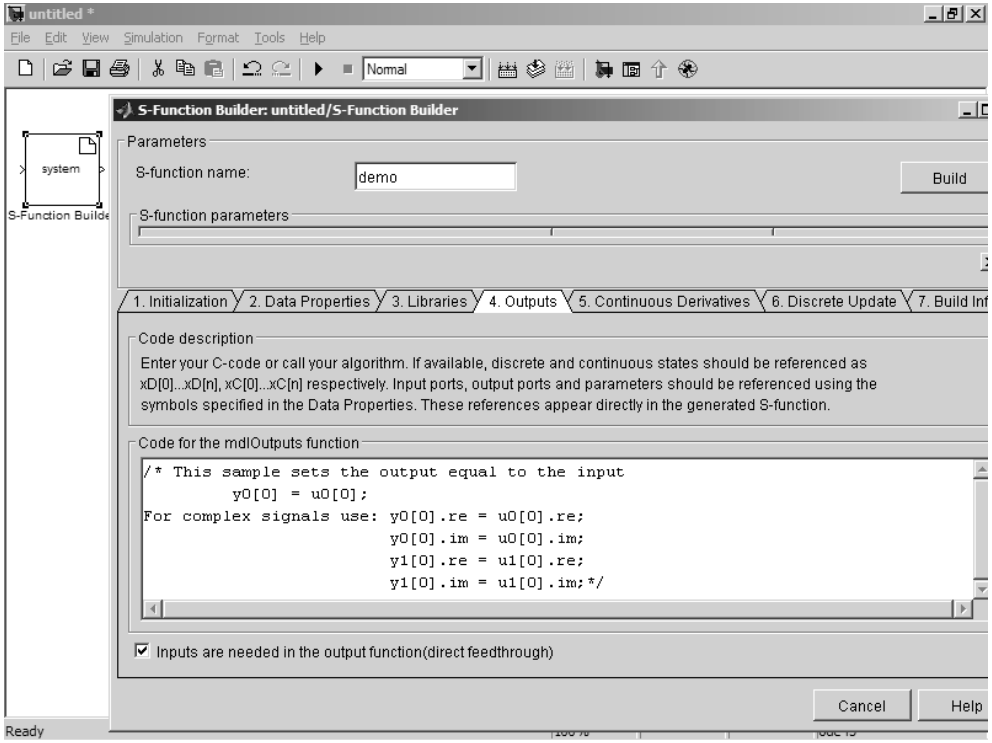


Рис. 6.44. Блок создания S-функции и его окно

Блоки этого раздела делятся на три категории:

- **Logic Operations** – блоки логических операций;
- **Bit Operations** – блоки побитовых операций;
- **Edge detections** – блоки детектирования пересечений.

Поскольку большинство из этих блоков уже было описано, повторять это описание не имеет смысла. Заметим, что назначение данных блоков вполне очевидно из их названий.

6.5.3. Блоки раздела Additional Math & Discrete

Еще один новый раздел библиотеки блоков Simulink 6 – это **Additional Math & Discrete**. Он содержит два подраздела, представленные на рис. 6.47.

Нетрудно заметить, что этот раздел имеет два подраздела:

- `Additional Math` – блоки дополнительных математических операций (в основном для реализации приращений);
- `Discrete` – дискретные блоки единичной задержки разного типа.

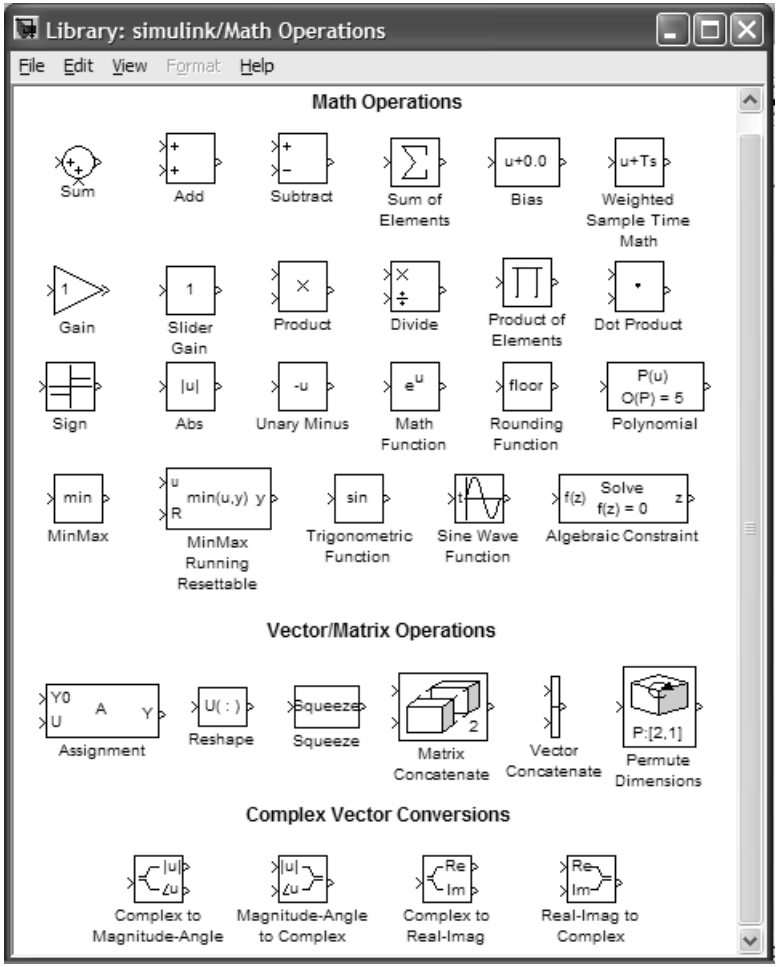


Рис. 6.45. Блоки раздела библиотеки Math Simulink 6.6/7

Надо прямо сказать, что блоки этих разделов применяются довольно редко, а их назначение вполне очевидно.

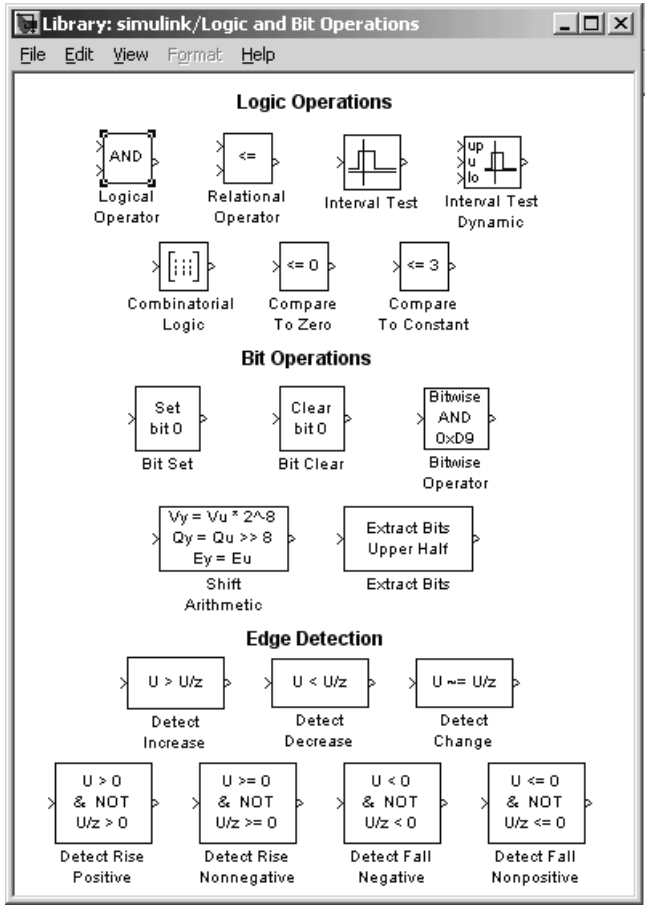


Рис. 6.46. Блоки раздела библиотеки Simulink 6 Logic and Bit Operations

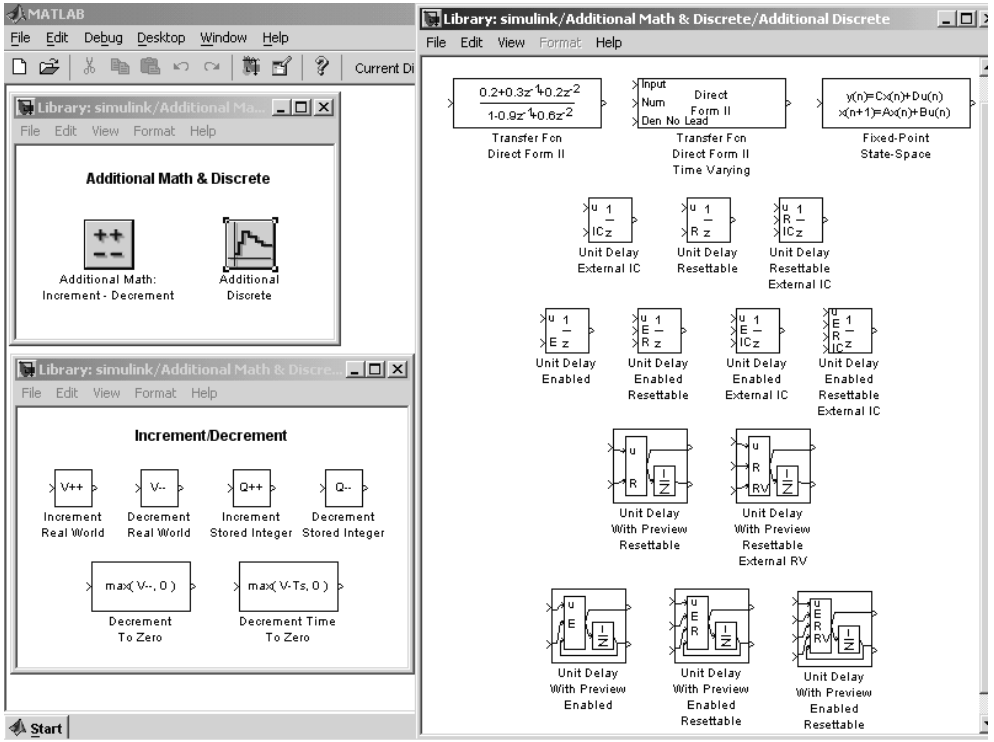


Рис. 6.47. Блоки раздела библиотеки Simulink 6 Additional Math & Discrete

Урок 7

Нелинейные, дискретные и специальные блоки

Многие системы и устройства содержат нелинейные, дискретные и иные специальные блоки. Такие блоки входят и в состав библиотек блоков Simulink. Они изучаются в данном уроке.

7.1. Нелинейные блоки

7.1.1. 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).

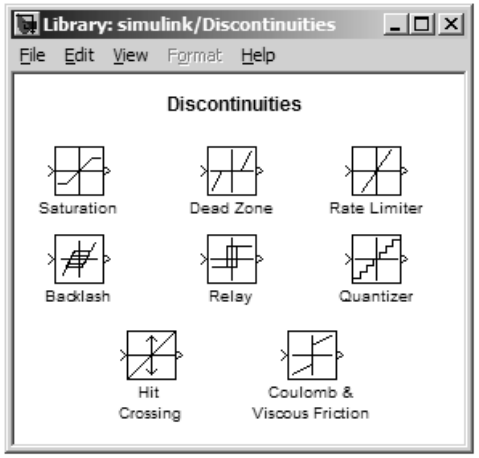


Рис. 7.1. Окно библиотеки с нелинейными компонентами

Среди нелинейных блоков следует отметить блоки с типичными нелинейностями, например блоки с характеристиками в виде типовых 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.

7.1.2. Блок ограничения Saturation

Блок **Saturation** представляет собой нелинейное устройство – идеальный ограничитель, 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).

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

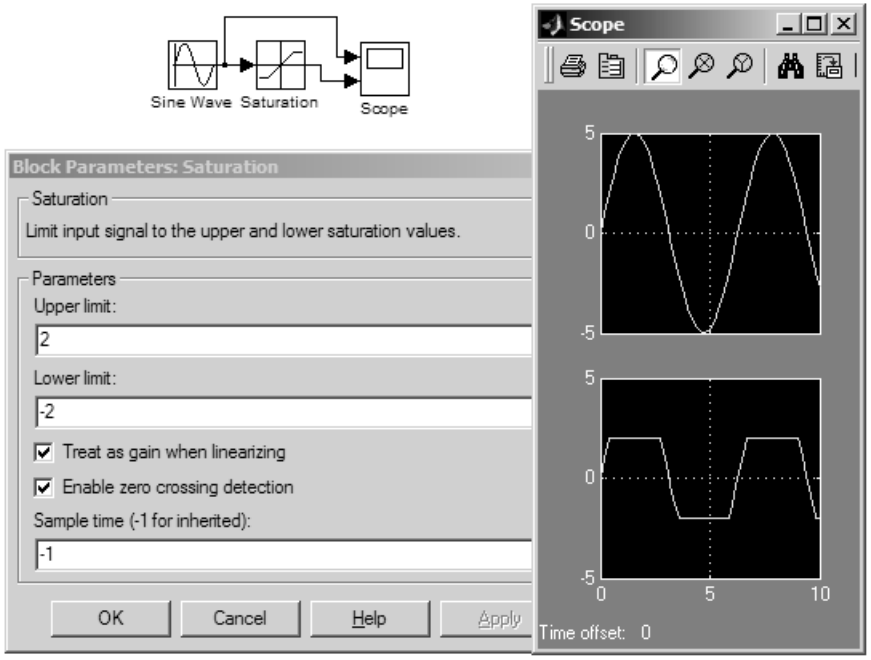


Рис. 7.2. Блок ограничения и окно установки его параметров

7.1.3. 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).

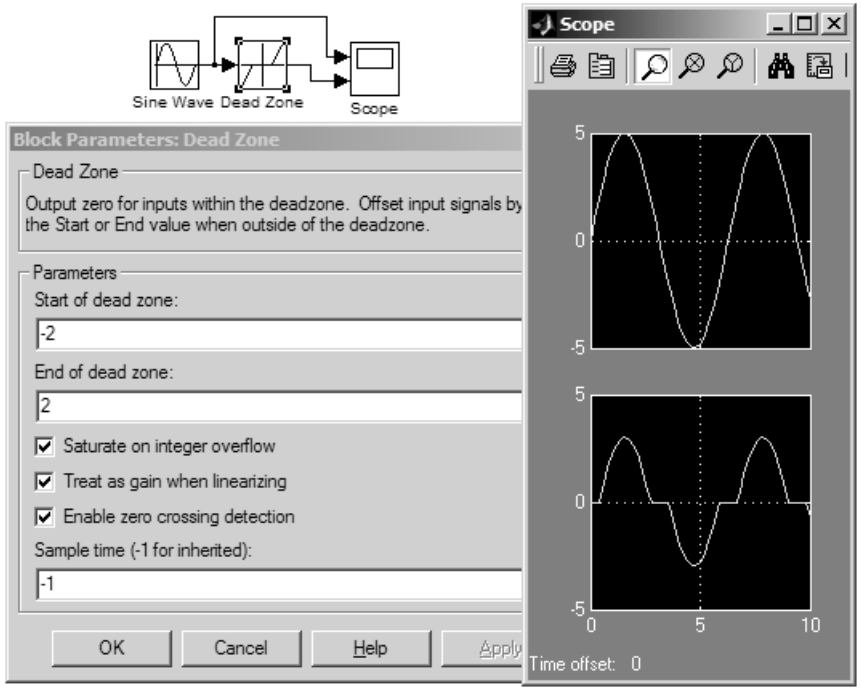


Рис. 7.3. Блок с зоной нечувствительности и окно установки его параметров

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.

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

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

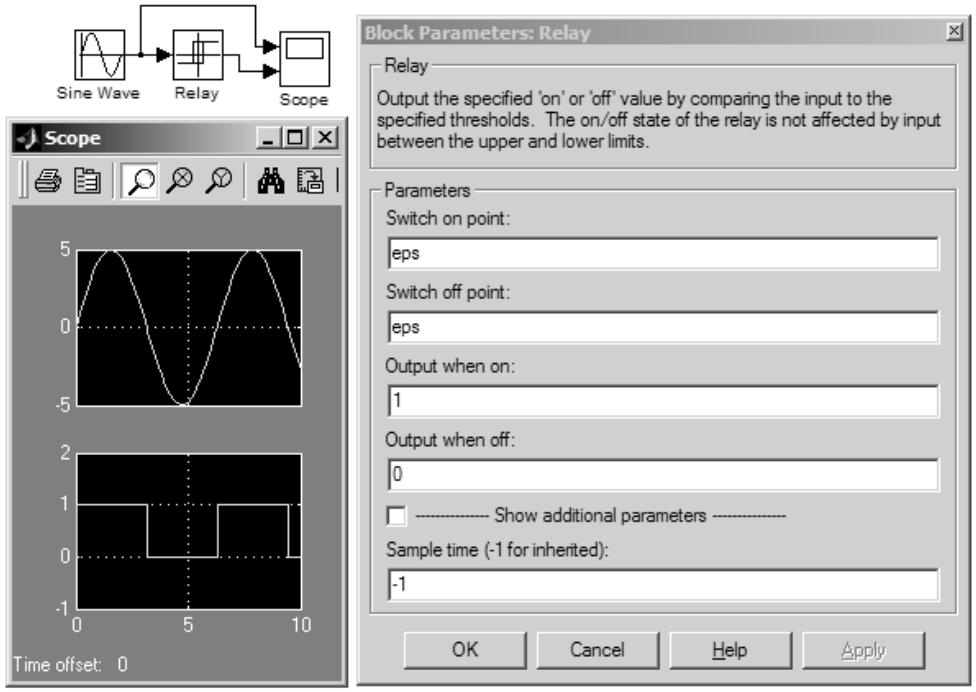


Рис. 7.4. Релейный блок и окно установки его параметров

7.1.5. Блок с ограничением скорости Rate Limiter

Блок с ограничением скорости **Rate Limiter** старается отслеживать входной сигнал в условиях задания ограничений на скорость нарастания и спада сигнала выходного сигнала блока (рис. 7.5).

Для вычисления скорости изменения сигнала используется соотношение

$$\text{rate} = [\boldsymbol{U}_i - \boldsymbol{OUT}_{i-1}]/[\boldsymbol{T}_i - \boldsymbol{T}_{i-1}],$$

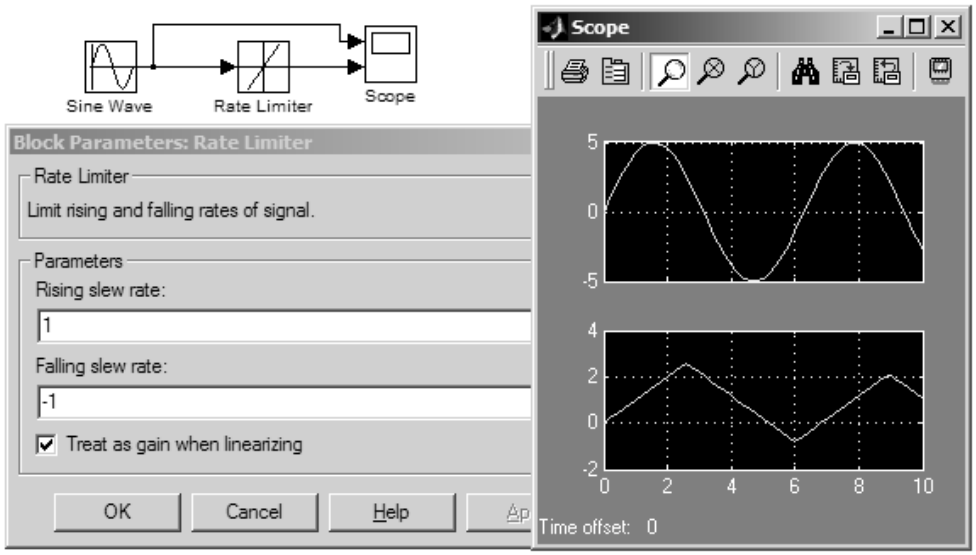


Рис. 7.5. Блок ограничения скорости и окно установки его параметров

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.

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

$$OUT_i = dT^*F + OUT_{i-1}.$$

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.

7.1.6. Блок квантования Quantizer

Блок **Quantizer** служит для квантования меняющихся сигналов с одинаковым шагом по уровню (рис. 7.6). Сигналы квантуются по уровню и превращаются в ступенчатые сигналы.

Блок имеет единственный параметр – шаг по уровню (по умолчанию 0,5). Рисунок 7.6 показывает квантование синусоидального сигнала. Можно отметить,

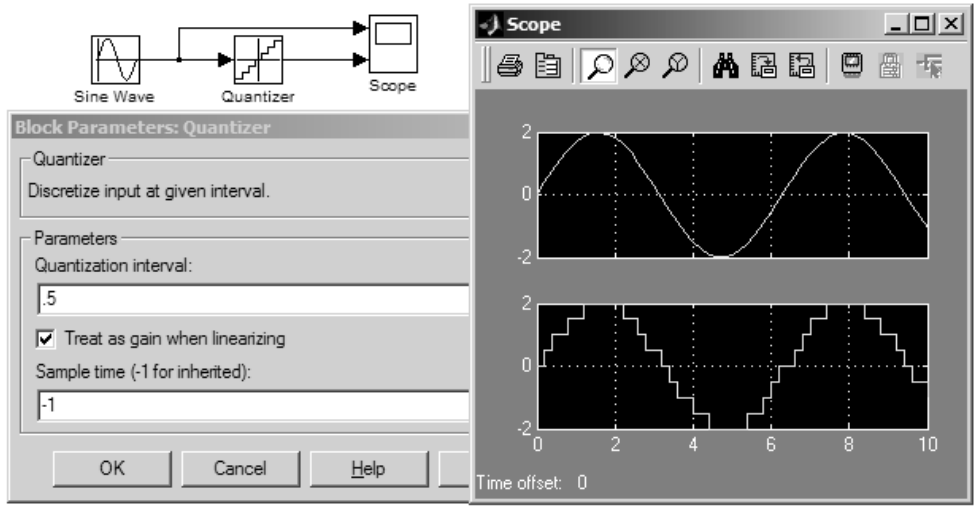


*Рис. 7.6. Блок **Quantizer** и окно установки его параметров*

что при большом шаге квантования его трудно назвать идеальным – точного слежения за уровнем входного сигнала нет.

7.1.7. Блок фрикционных эффектов Coulombic and Viscous Friction

Блок фрикционных эффектов **Coulombic and Viscous Friction** служит для моделирования фрикционных эффектов сухого и вязкого трения (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.

7.1.8. Блок люфта Backlash

Блок **Backlash** имитирует эффект возникновения люфта (рис. 7.8). Этот эффект создает передаточную характеристику гистерезисного типа, которая представляется графически в пиктограмме блока.

Блок имеет два параметра: ширину 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.

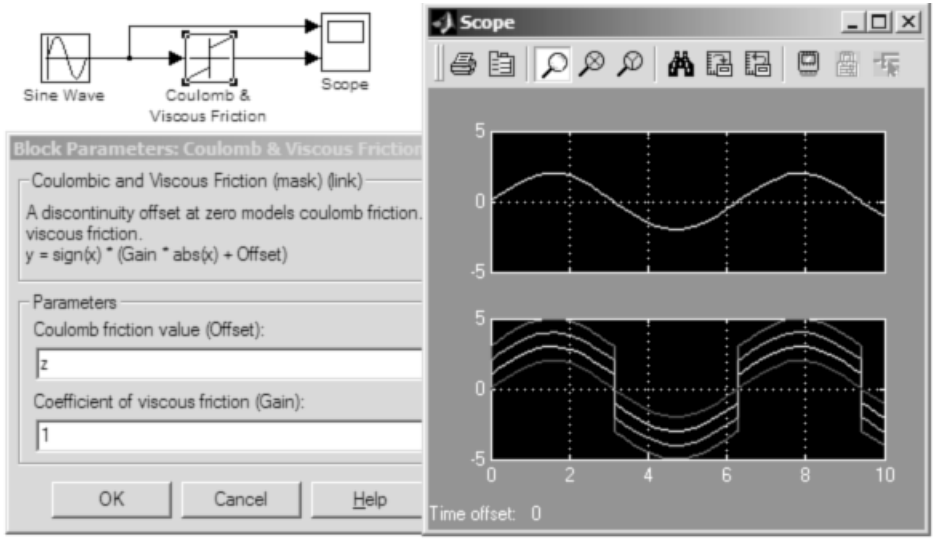


Рис. 7.7. Блок фрикционных эффектов и окно установки его параметров

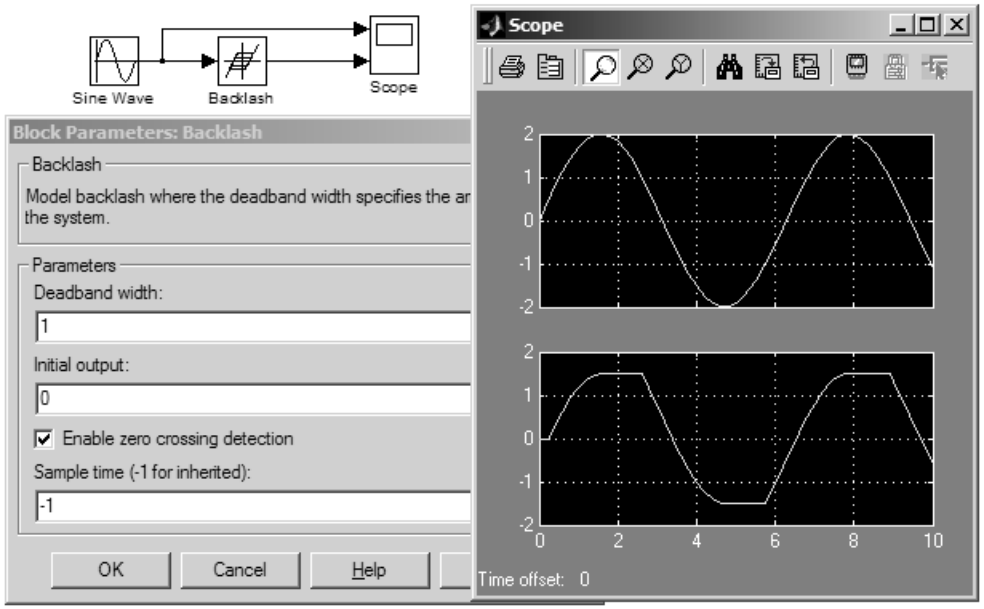


Рис. 7.8. Блок люфта и окно установки его параметров

7.1.9. Детектор пересечения заданного уровня Hit Crossing

Блок **Hit Crossing** (рис. 7.9) фиксирует прохождение сигналом заданного уровня (по умолчанию нулевого) и при каждом пересечении вырабатывает короткий импульс единичной амплитуды.

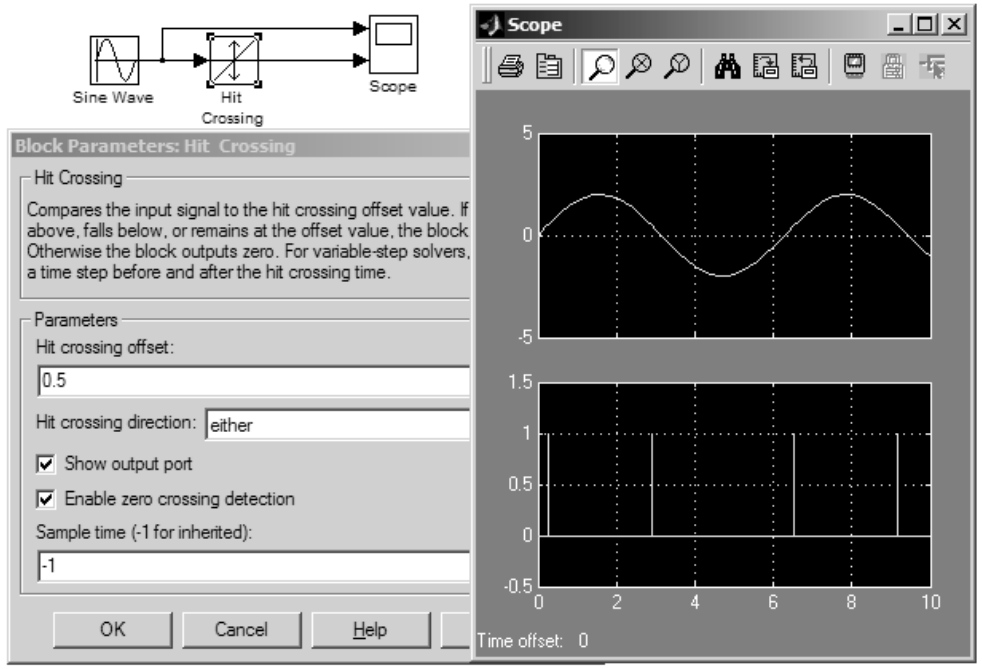


Рис. 7.9. Блок пересечения заданного уровня и окно установки его параметров

Окно установки параметров данного блока позволяет задать следующие параметры:

- `Hit crossing offset` – порог пересечения;
- `Hit crossing directions` – направление пересечения (`either` – любое, `rising` – нарастание, `failing` – при спаде);
- `Show output port` – показать выходной порт;
- `Enable zero crossing detection` – фиксировать прохождение через нуль.

7.2. Дискретные блоки

7.2.1. 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. 7.10 представлено окно библиотеки **Discrete** с этими устройствами.

Некоторые дискретные блоки имеют передаточные функции, напоминающие уже описанные для непрерывных блоков. 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.

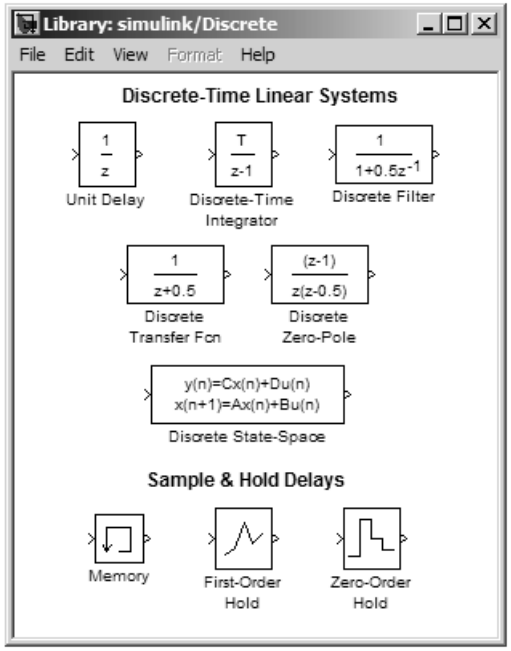


Рис. 7.10. Раздел библиотеки с дискретными блоками

7.2.2. Блок дискретной единичной задержки Unit Delay

Блок **Unit Delay** обеспечивает задержку входного сигнала на один шаг эталонного времени. Это иллюстрирует рис. 7.11, на котором демонстрируется задержка перепада.

Окно параметров блока, показанное также на рис. 7.11, предусматривает установку начального значения для уровня выходного сигнала **Initial condition** и эталонного времени `Simple Time`.

7.2.3. 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. 7.12).

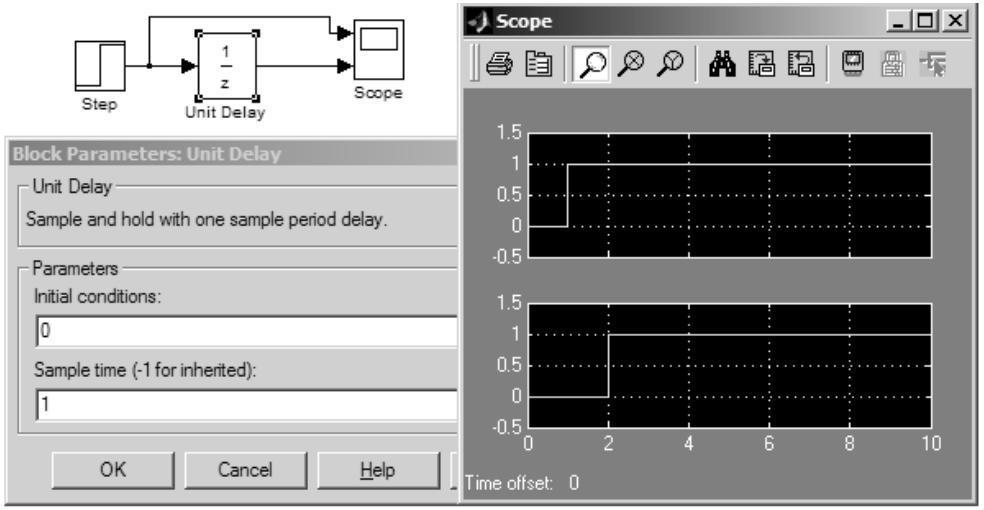


Рис. 7.11. Блок дискретной единичной задержки

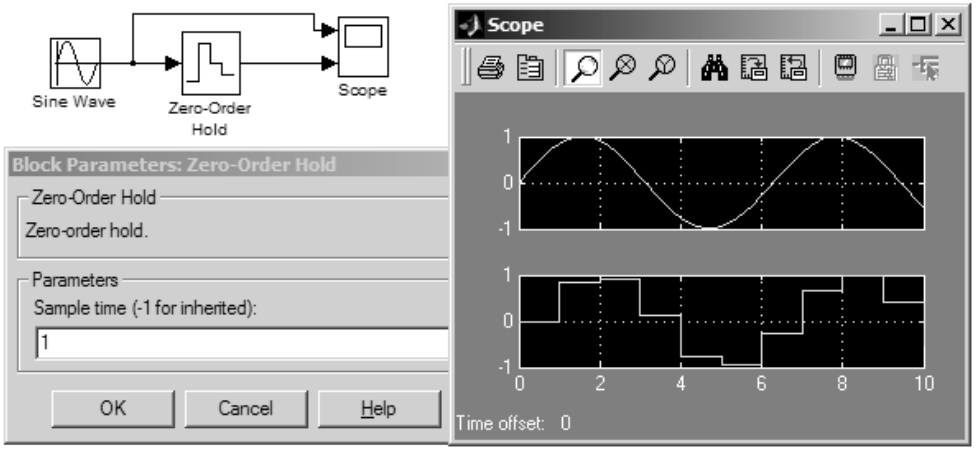


Рис. 7.12. Блок дискретного экстраполятора нулевого порядка

Предусмотрена установка единственного параметра – эталонного времени `Simple Time`.

7.2.4. Блок экстраполятора первого порядка First-Order Hold

Блок экстраполятора первого порядка **First-Order Hold** задерживает выходной сигнал на заданный промежуток времени и задает линейное 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. 7.13.

В окне параметров этого блока (рис. 7.13) предусмотрена установка единственного параметра – эталонного времени (по умолчанию 1). Обратите внимание, что оно может быть дробным числом, как в приведенном примере.

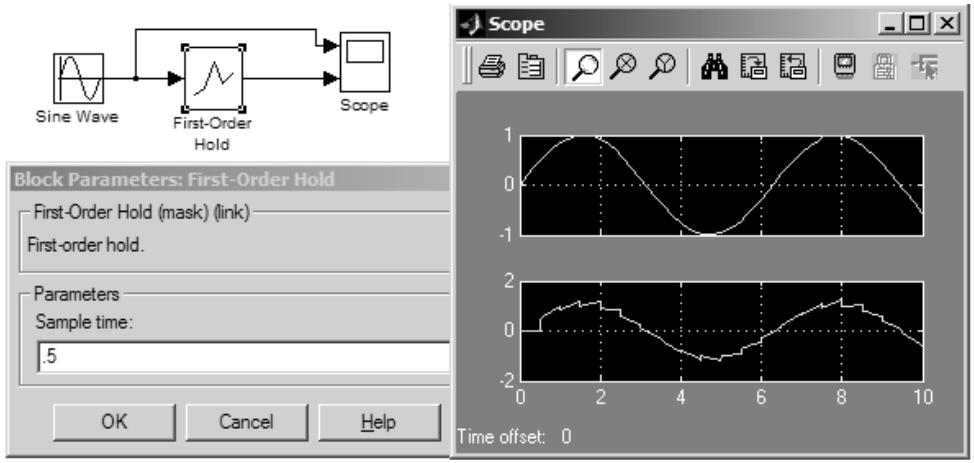


Рис. 7.13. Блок дискретного экстраполятора первого порядка

7.2.5. Блок дискретного интегратора времени Discrete-Time Integrator

Блок **Discrete-Time Integrator** служит для дискретного интегрирования времени (рис. 7.14). 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. 7.14. Основными являются следующие параметры:

- `Integration method` – метод численного интегрирования;
- `External reset` – сброс внешним сигналом;
- `Initial condition` – задание начальных значений выходного сигнала;
- `Limit output` – ввод ограничений на изменение выходного сигнала сверху `Upper` и снизу `Lower`;
- `Show saturation port` – показ порта, дающего сигнал об ограничении;
- `Show state port` – показ порта статуса интегратора;
- `Sample time` – эталонное время.

Метод интегрирования выбирается из раскрывающегося списка. Возможен выбор одного из трех методов:

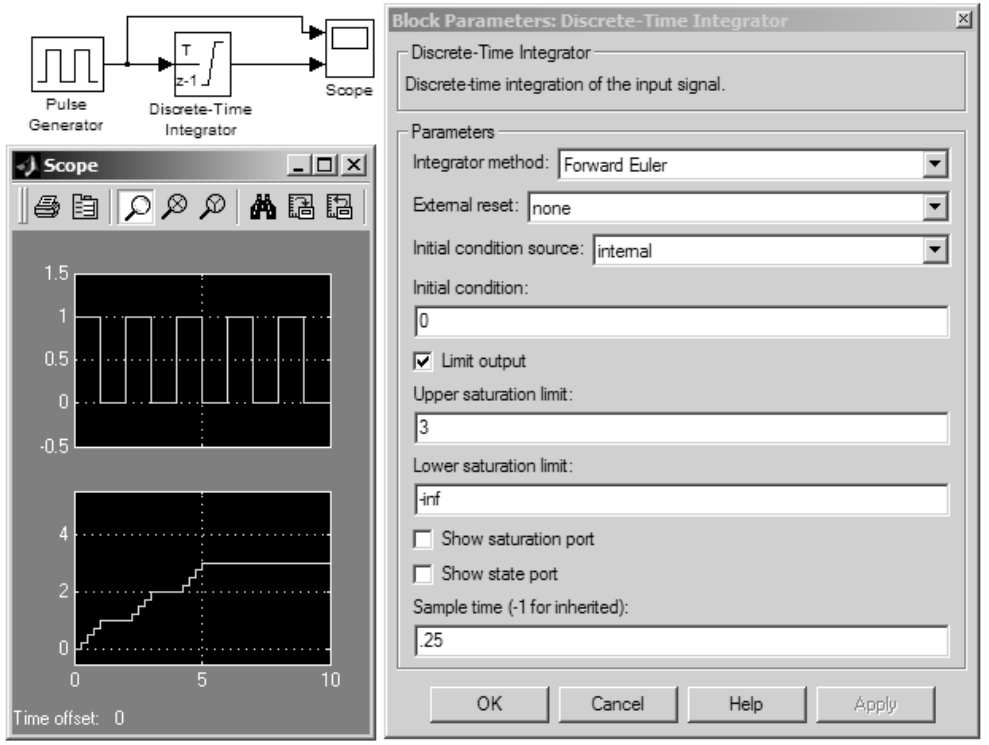


Рис. 7.14. Пример применения блока дискретного интегратора времени для интегрирования прямоугольных импульсов

- **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 знание таких соотношений для практического моделирования, как правило, не требуется (достаточно знать суть соответствующего метода). На рис. 7.14 надпись внутри графического изображения блока дана для методов Эйлера, для метода трапеций она иная.

7.2.6. 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. 7.15).

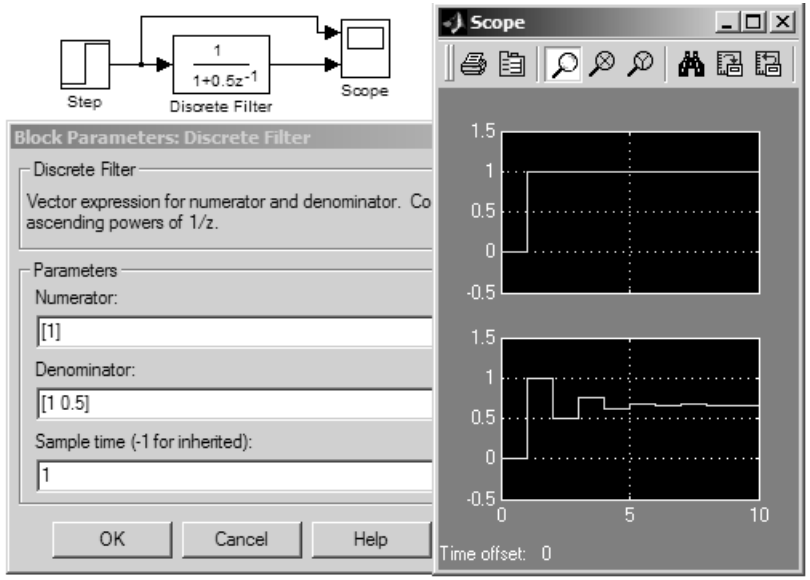


Рис. 7.15. Блок дискретного фильтра

Параметрами дискретного фильтра являются векторы, содержащие коэффициенты полиномов числителя и знаменателя, и эталонное время. Надпись в графическом представлении блока зависит от выбранного полинома.

7.2.7. Блок памяти Memory

Блок **Memory (Память)** запоминает входной сигнал и смещает его на один такт времени (рис. 7.16).

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.

7.2.8. Блок дискретной передаточной функции Discrete Transfer Fcn

Блок **Discrete Transfer Fcn** служит для задания дискретной передаточной функции. Как и подобная непрерывная функция, она задается коэффициентами полиномов числителя и знаменателя этой функции переменной z. Рисунок 7.17 демонстрирует применение данного блока.

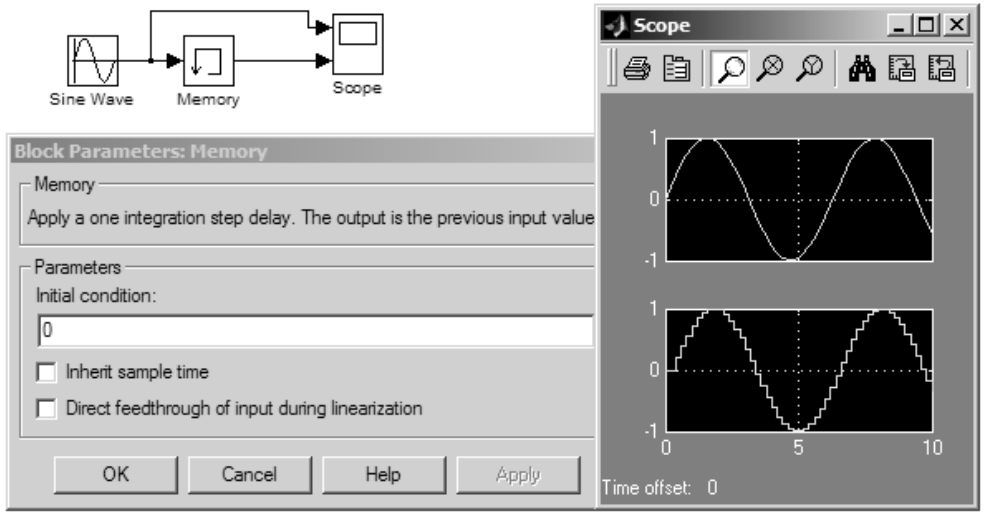


*Рис. 7.16. Применение блока **Memory***

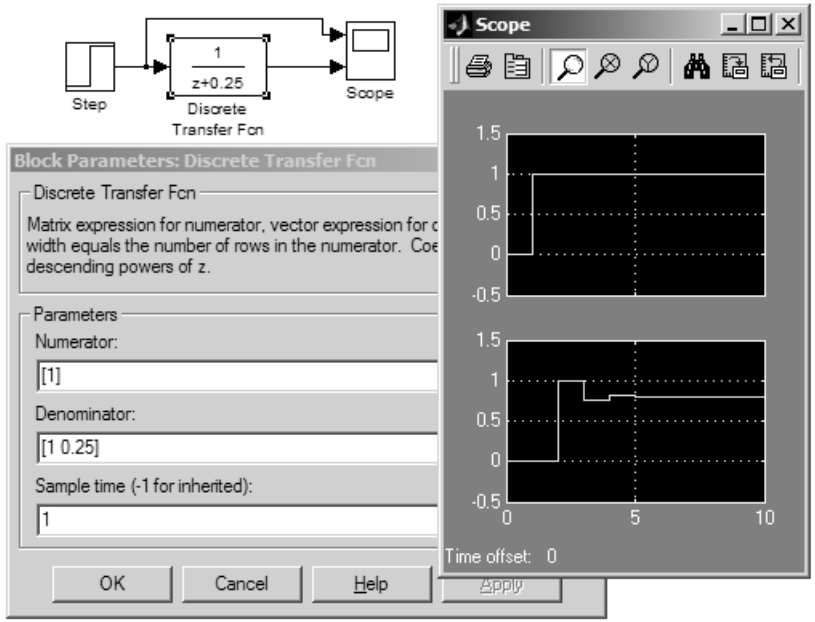


*Рис. 7.17. Применение блока **Memory Discrete Transfer Fcn***

7.2.9. Блок задания дискретной функции Discrete Zero Pole

Блок **Discrete Zero Pole** служит для задания дискретной передаточной функции с заданными нулями и полюсами. Пример применения блока и окно установки его параметров представлены на рис. 7.18. Параметры этого блока аналогичны описанным для такого непрерывного блока.

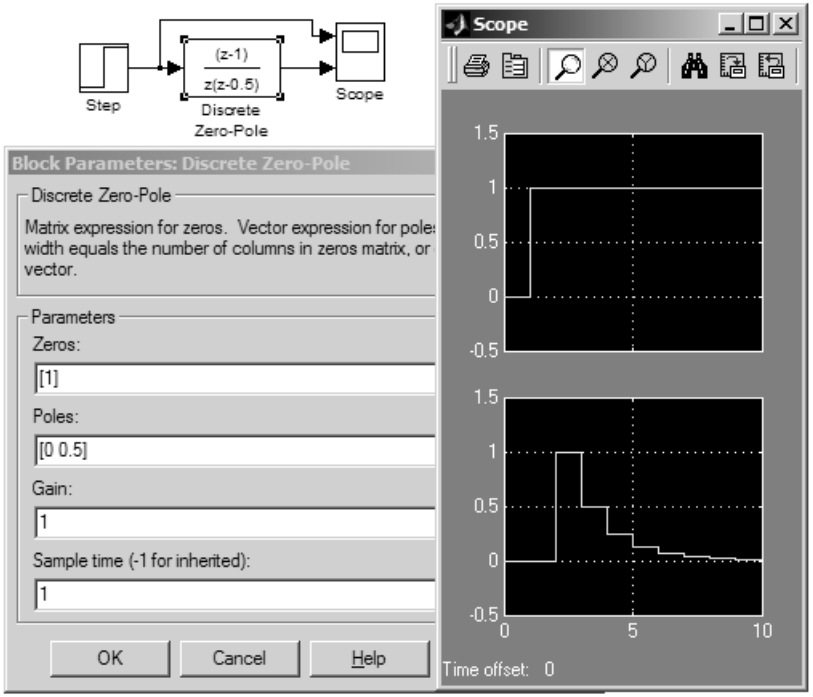


Рис. 7.18. Применение блока ***Discrete Zero Pole***

7.2.10. Блок Discrete State Space

Блок **Discrete State Space** служит для формирования дискретного пространства состояний. Пример применения блока и окно установки его параметров показаны на рис. 7.19.

Параметры блока те же, что у подобного непрерывного блока.

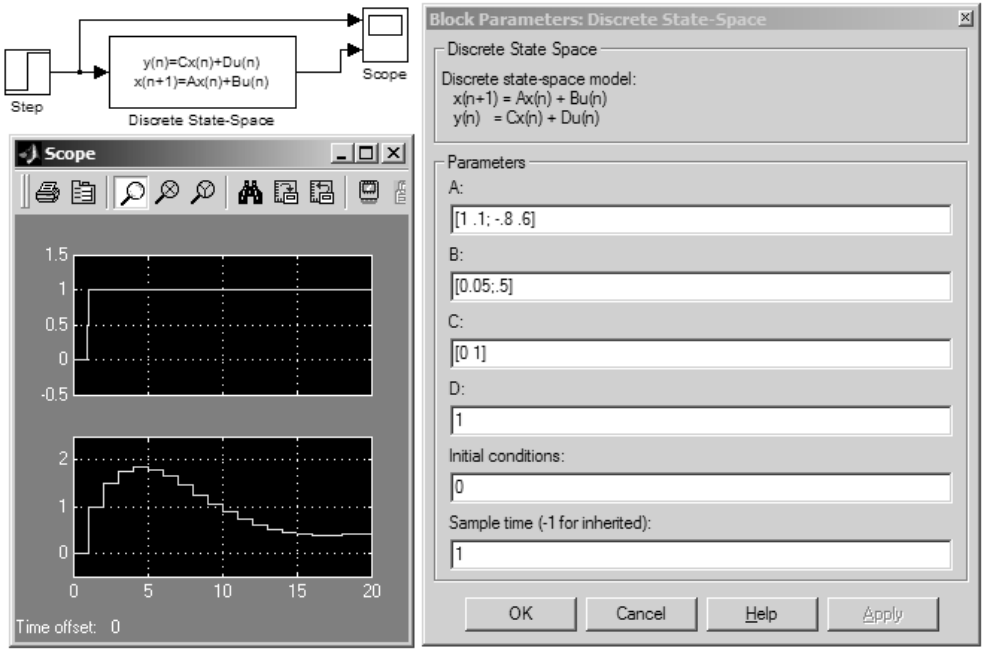


Рис. 7.19. Применение блока ***Discrete State Space***

7.3. Библиотеки Simulink Extras

7.3.1. 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. Данная библиотека содержит наборы блоков с более широкими функциями, чем рассмотренные ранее в разделах 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. 7.20. Работа с этой библиотекой ничем не отличается от работы с основной библиотекой.

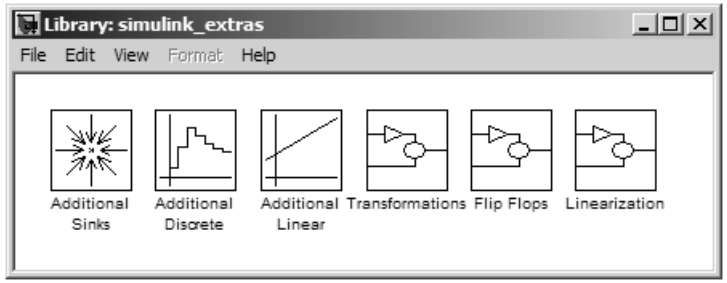


*Рис. 7.20. Раздел библиотеки **Simulink Extras***

Как видно из рис. 7.20, 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 (в Simulink 5 удалена в связи с переносом в раздел демонстрационных примеров).

7.3.2. 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. 7.21). Их единственным отличием от описанных ранее блоков является возможность инициализации входов и состояний.

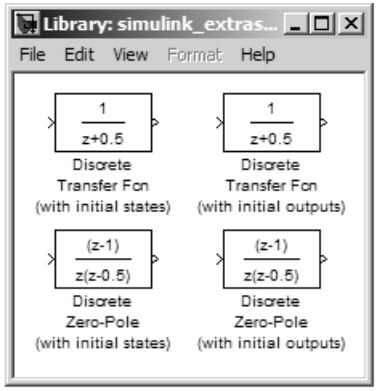


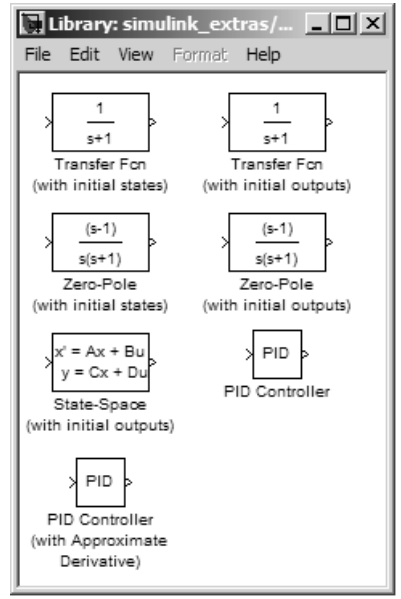
*Рис. 7.21. Раздел библиотеки **Simulink Extras** с дискретными блоками*

7.3.3. Дополнительные линейные блоки

Состав дополнительных линейных блоков раздела **Additional Linear** показан на рис. 7.22.

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. Последние мы рассматривать не будем ввиду

*Рис. 7.22. Раздел библиотеки **Simulink Extras** с дополнительными линейными блоками*



идентичности их с подобными блоками, описанными ранее. Остановимся на реализации блоков PID-контроллеров.

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. 7.23 дан пример применения PID-контроллера при входном сигнале в виде прямоугольных импульсов. Выходной сигнал задан как сумма входного

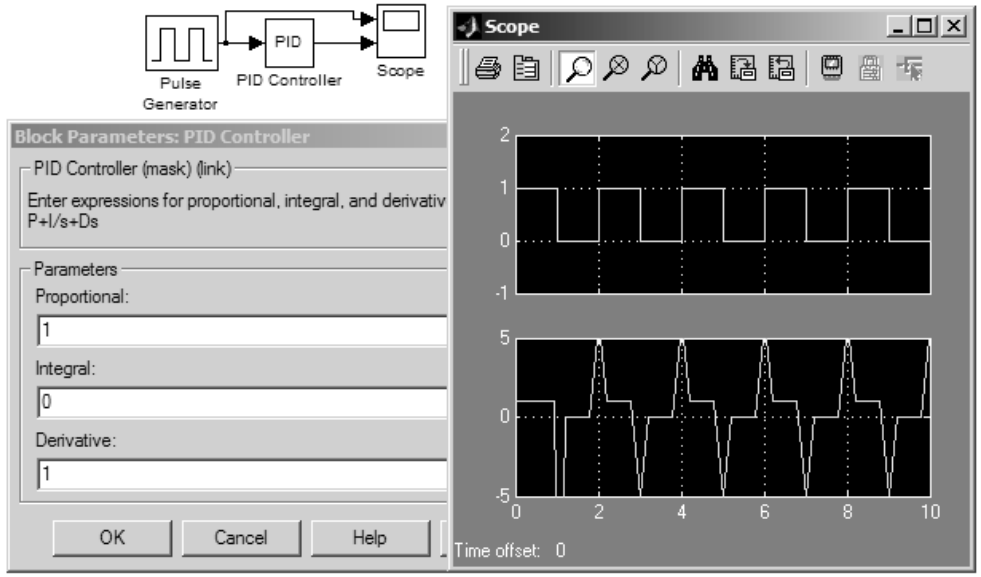


Рис. 7.23. Пример применения PID-контроллера

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.

Другой блок – **PID controller with Approximate Derivative** (PID-контроллер с улучшенным вычислением производной) вычисляет выходной сигнал как

$$OUT = \mathbf{3} + I/s + Ds/(1/Ns + 1).$$

За счет применения дополнительного параметра ***N*** улучшается вычисление производной.

Пример, показанный на рис. 7.24, 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.

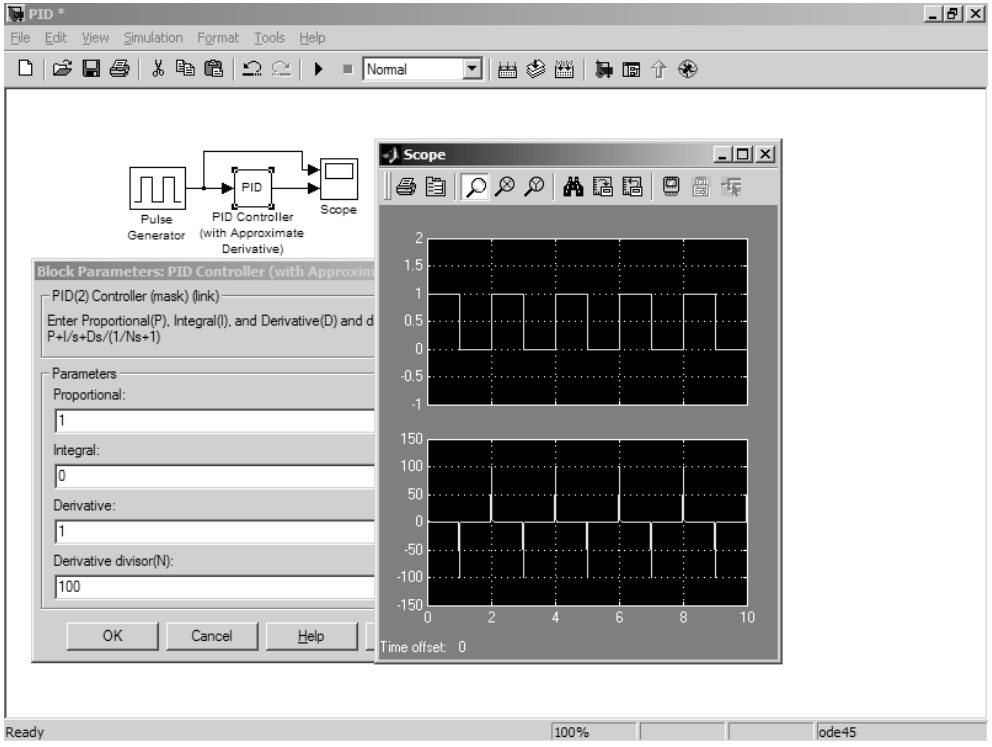


Рис. 7.24. Пример применения улучшенного PID-контроллера при обработке сигнала в виде прямоугольных импульсов

Нетрудно заметить, что в данном случае дифференцирование входных прямоугольных импульсов хотя тоже не идеально, но намного лучше, чем в примере, представленном на рис. 7.23.

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.

7.3.4. Дополнительные блоки Additional Sinks

Раздел дополнительных блоков **Additional Sinks** содержит ряд новых виртуальных регистраторов (рис. 7.25).

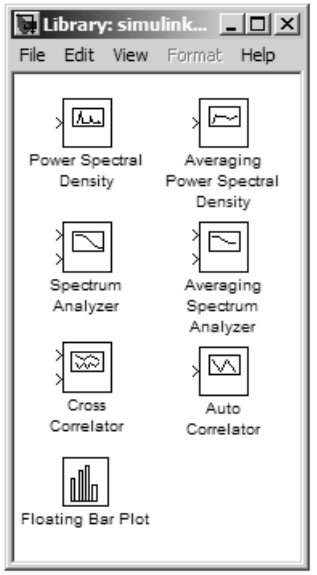


*Рис. 7.25. Раздел дополнительных блоков **Additional Sinks***

В него входят следующие блоки:

- **Auto Correlator** – автоматический коррелятор (используется с пакетом **Signal Processing Toolbox**);
- **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 гистограмм;
- **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).

7.3.5. Блоки спектрального анализа

Рассмотрим, как наиболее характерный, блок **Averaging Power Spectral Density**. Он служит для наглядного представления формы сигналов и их энергетического спектра (рис. 7.26). 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;

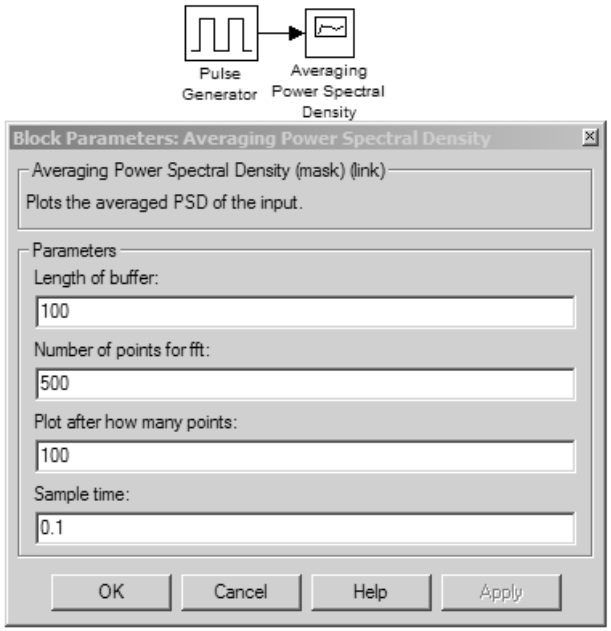


Рис. 7.26. Пример применения блока ***Averaging Power Spectral Density***

- `Plot after how many points` – построение графика после задания числа точек;
- `Sample time` – эталонное время.

На рис. 7.27 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 отрезками прямых. 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).

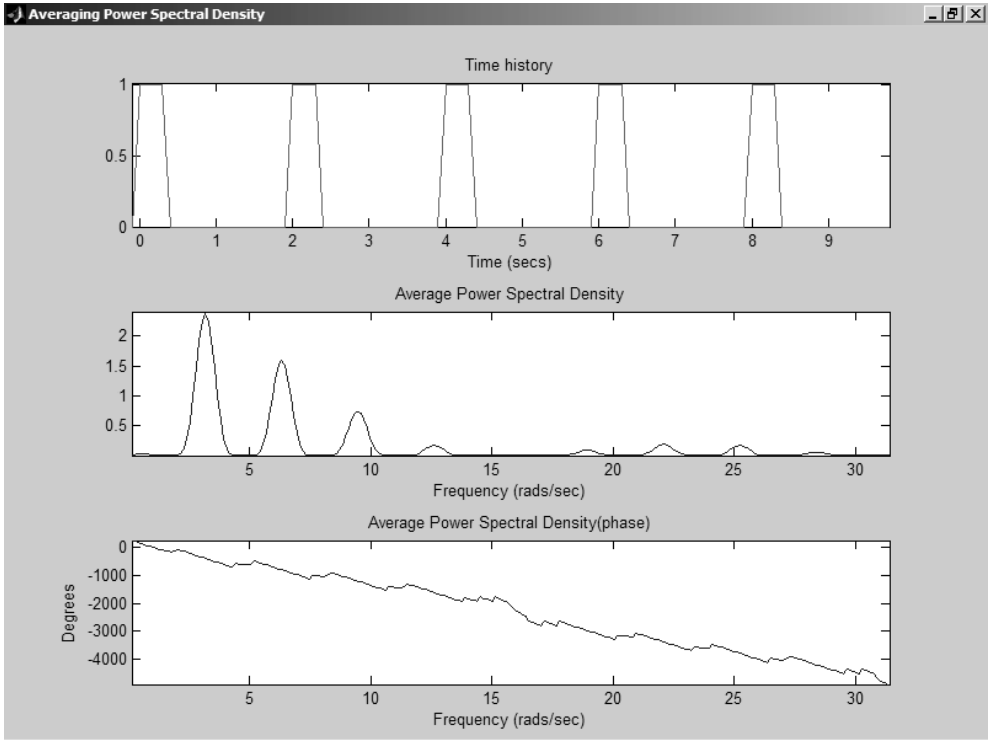


Рис. 7.27. Графики, создаваемые блоком ***Averaging Power Spectral Density***

Остальные анализаторы спектра используются совершенно аналогично и имеют такое же окно представления результатов спектрального анализа. Их различия вытекают из наименования блоков (см. выше).

7.3.6. 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. Напомним, что если оба сигнала меняются идентично, то коэффициент корреляции равен 1, а его значение 0 означает различное изменение сигналов. Рисунок 7.28 показывает применение этого блока в случае, когда на входе действуют сигналы разного вида (синусоида и пилообразный импульс) с одинаковыми частотами.

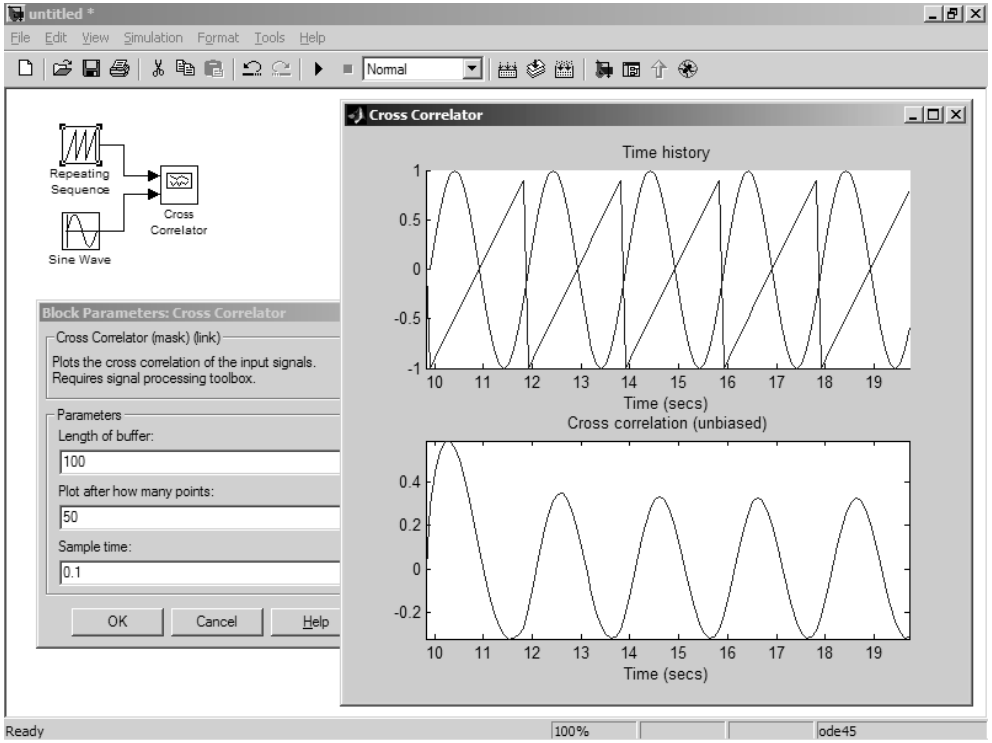


Рис. 7.28. Пример применения блока кросс-коррелятора

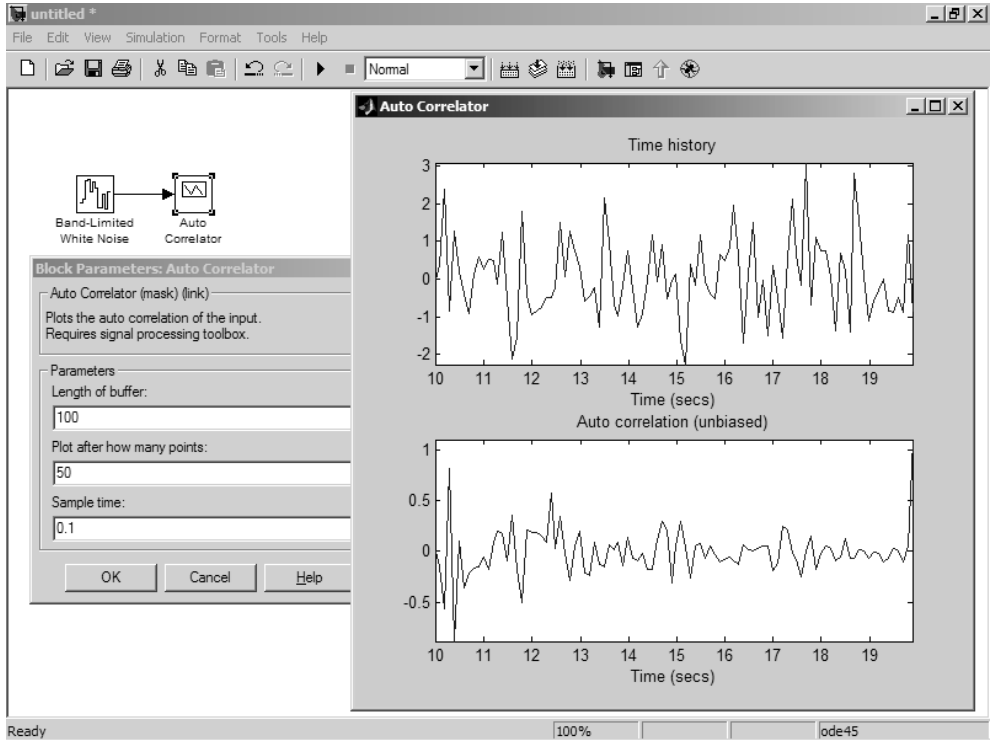


Рис. 7.29. Пример применения блока автокоррелятора

Параметры этого блока имеют то же назначение, что и одноименные параметры блока **Averaging Power Spectral Density**.

7.3.7. Блок автокоррелятора Cross-Correlator

Блок автокоррелятора служит для вычисления автокорреляционной функции сигналов. Пример его применения и окно установки параметров блока представлены на рис. 7.29. В данном примере вычисляется автокорреляционная функция белого шума – она, испытывая случайные флуктуации, приближается к нулю.

7.3.8. Обзор раздела библиотеки Flip Flops

Раздел библиотеки **Simulink Extras Flip Flops** содержит генератор тактовых прямоугольных импульсов и четыре блока триггеров (рис. 7.30).

Все они описаны ниже.

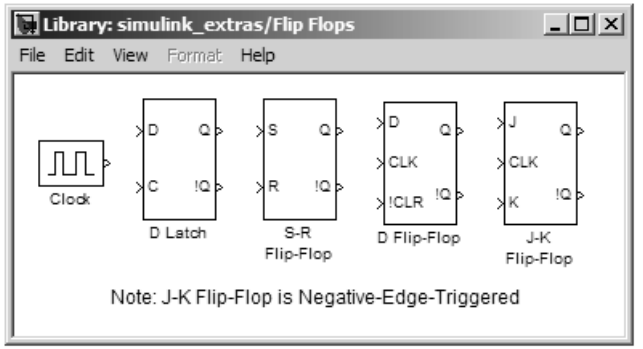


*Рис. 7.30. Раздел библиотеки **Flip Flops***

7.3.9. 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. 7.31.

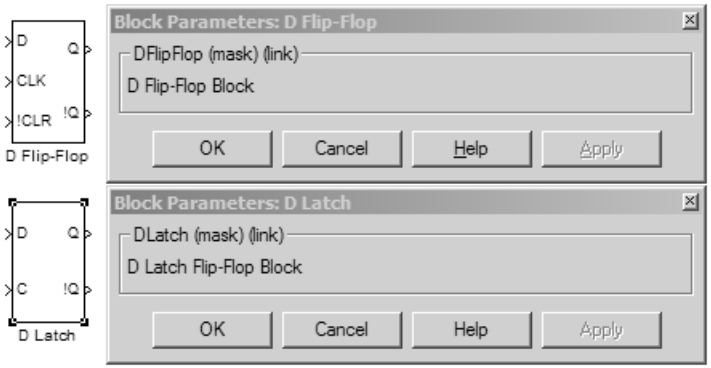


*Рис. 7.31. Пример применения блока **Clock***

Единственным параметром этого источника импульсов является период (в тактах периода эталонного времени). Этот источник обычно используется для тактирования работы триггеров.

7.3.10. Триггерные блоки

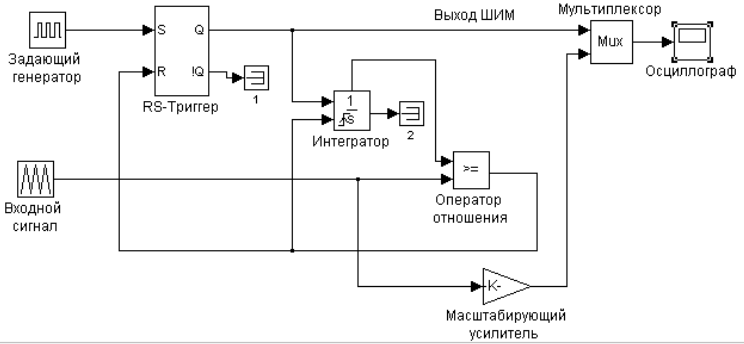
Раздел библиотеки **Flip Flops** содержит четыре блока триггерных устройств, реализующих следующие типы стандартных триггерных устройств: **D Flip Flop**, **D Latch**, **J-K Flip-Flops** и **S-R Flip-Flops**. На рис. 7.32 показаны блоки D-триггеров и окна установки их параметров. Эти триггеры не имеют каких-либо устанавливаемых параметров.

Для JK и RS-триггеров в окне установки параметров (рис. 7.32) задается единственный параметр – уровень инициализации.

7.3.11. Пример построения широтно-импульсного модулятора

Триггеры часто используются для построения импульсных устройств. На рис. 7.33 дан пример построения модели широтно-импульсного модулятора (ШИМ) на основе RS-триггера. Названия блоков в этом примере даны на русском языке.

*Рис. 7.32. Триггерные блоки **D Flip Flop** и **D Latch** и окна установки их параметров*



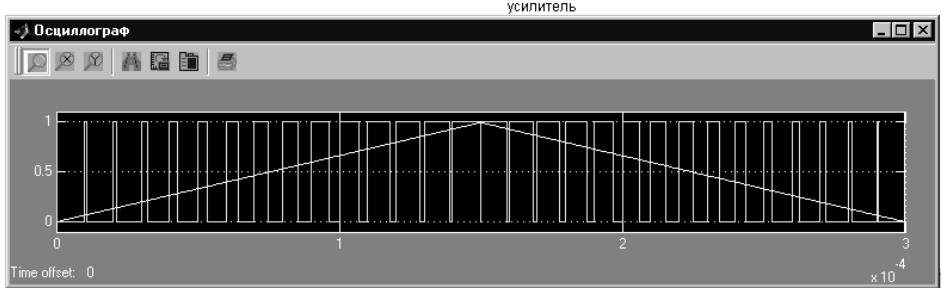


Рис. 7.33. Пример построения модели широтно-импульсного модулятора

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**, кроме того, должен быть установлен флажок **Show state port**. Подобная установка внешнего сброса задает у входа блока интегратора особый знак переключения и задает 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.

7.3.12. Раздел Linearization

Раздел **Linearization** содержит всего два блока – дифференцирующий блок и блок заданной временной задержки.

Работу дифференцирующего блока поясняет рис. 7.34. 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.

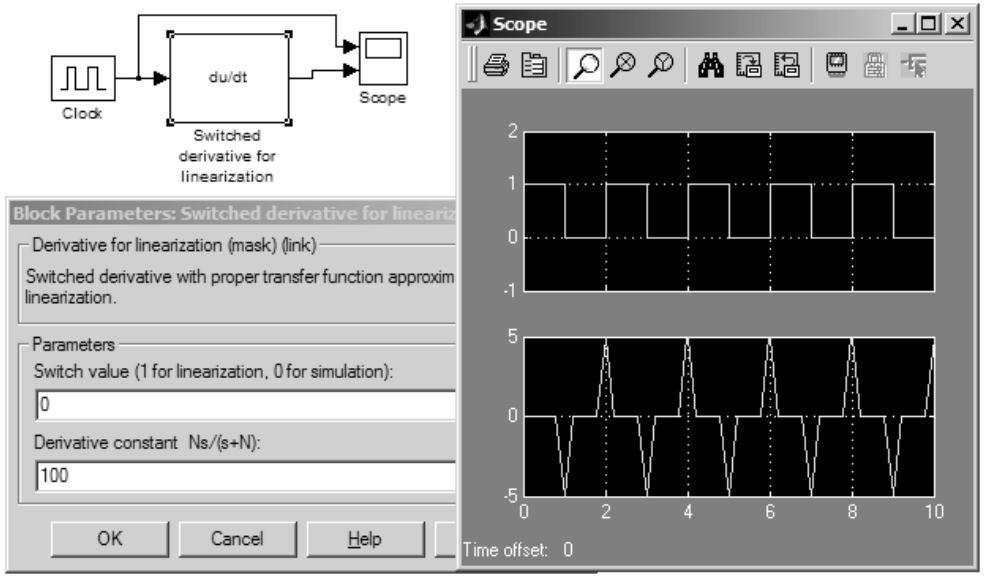


Рис. 7.34. Пример работы блока дифференцирования

Достаточно установить указанный параметр равным 1, и качество дифференцирования намного возрастает – см. пример на рис. 7.35.

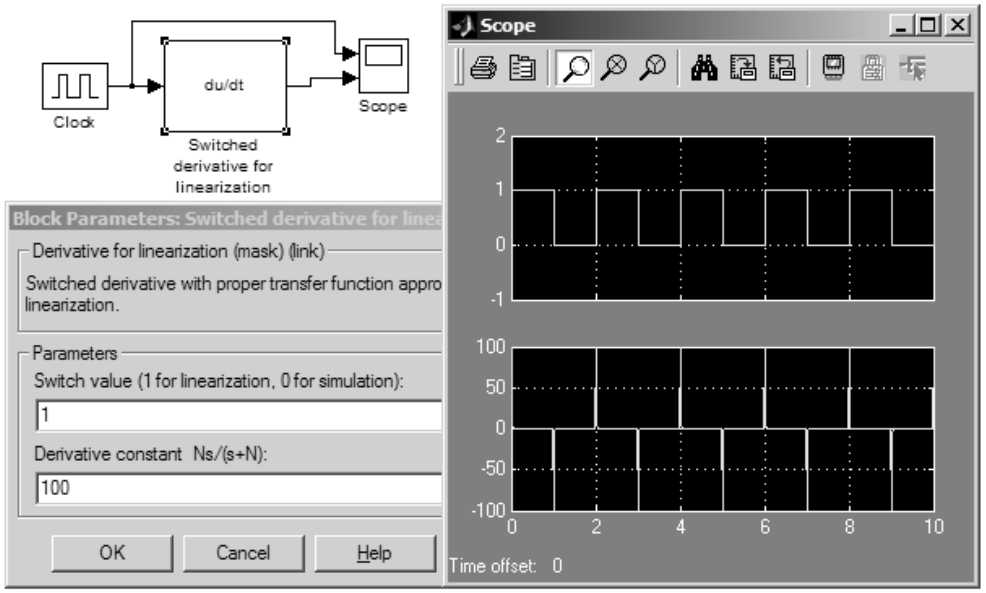


Рис. 7.35. Пример дифференцирования с повышенным качеством

Помимо указанного параметра, в этом блоке задается еще один параметр – константа дифференцирования (она входит в знаменатель операторной передаточной функции блока).

7.3.13. 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. 7.36.

В окне параметров временной задержки помимо параметра **Time delay** устанавливаются начальное значение входного сигнала **Initial input**, начальная емкость буфера **Initial buffer size** и порядок Паде-линеаризации.

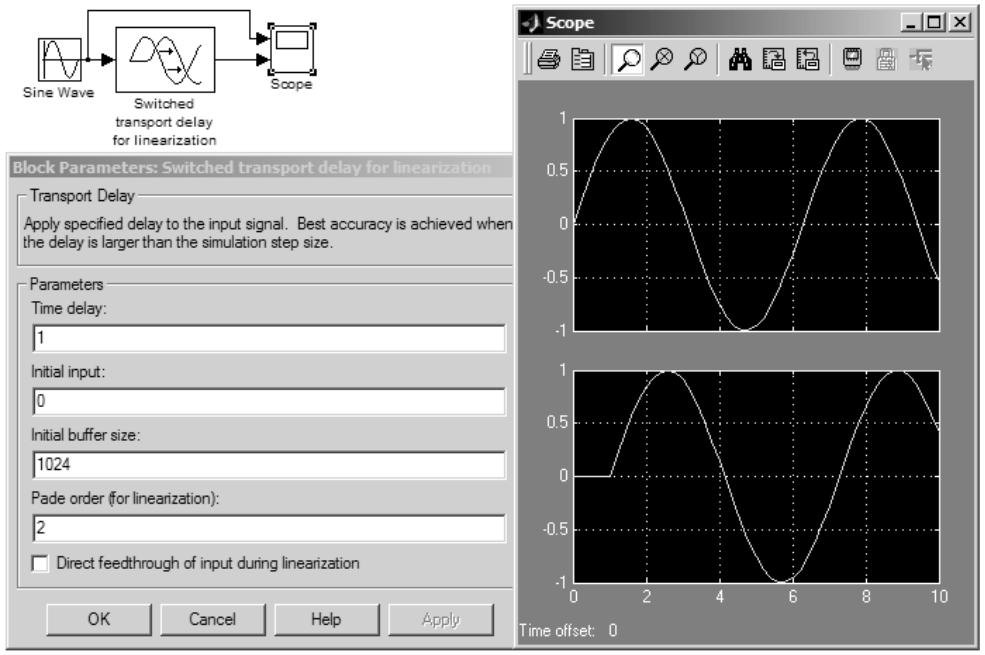


Рис. 7.36. Пример применения временной задержки

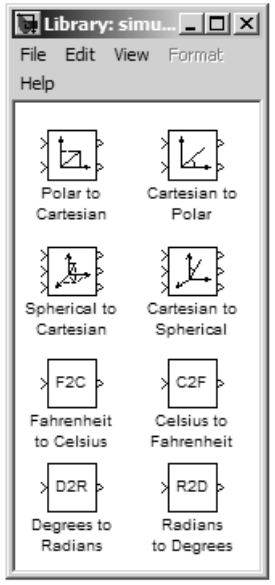


Рис. 7.37. Раздел библиотеки с блоками преобразований

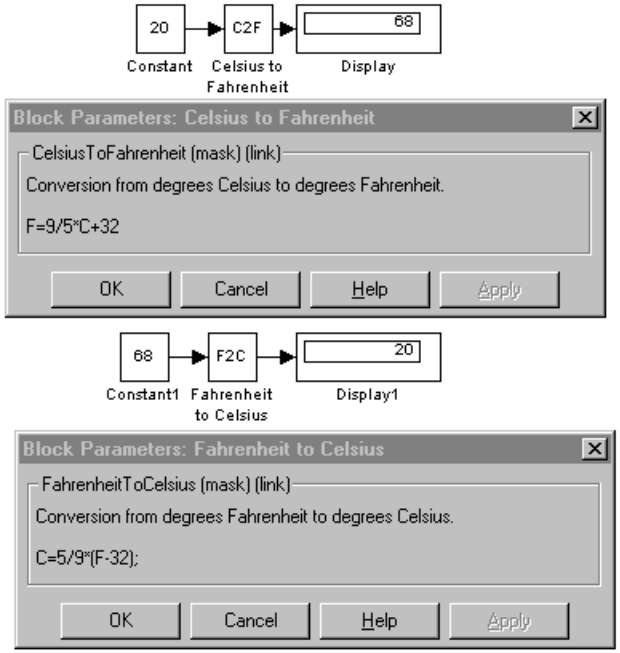


Рис. 7.38. Примеры преобразования температуры

7.4. Блоки преобразований

7.4.1. Обзор раздела преобразований Transformations

Раздел преобразований `Transformations` содержит восемь блоков для осуществления типичных преобразований – это преобразование температуры, углов и систем координат (рис. 7.37).

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.

7.4.2. 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. 7.38 сверху.

7.4.3. Блок преобразования температуры Fahrenheit to Celsius

Работу блока **Fahrenheit to Celsius**, переводящего температуру из градусов Фаренгейта в градусы Цельсия, показывает также рис. 7.38 (снизу).

7.4.4. 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. 7.39 (сверху).

7.4.5. 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. 7.39 (снизу).

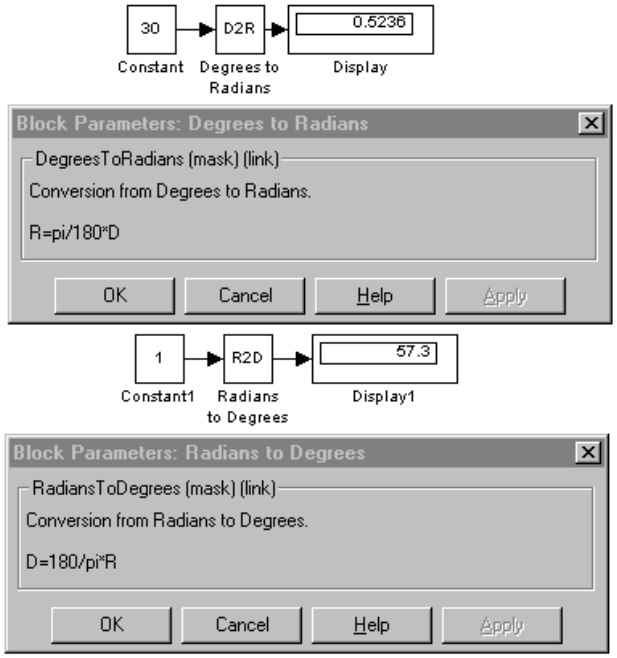


Рис. 7.39. Примеры преобразования углов

7.4.6. Блок преобразования координат Cartesian to Polar

Блок **Cartesian to Polar** служит для преобразования прямоугольных координат точки на плоскости в полярные координаты (рис. 7.40).

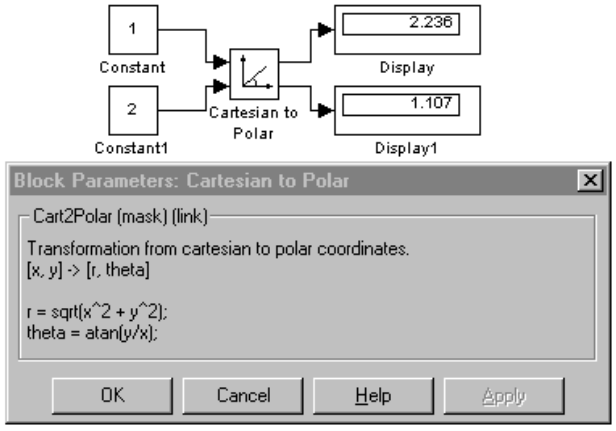


Рис. 7.40. Пример преобразования прямоугольных координат точки на плоскости в полярные координаты

7.4.7. Блок преобразования координат Polar to Cartesian

Блок **Polar to Cartesian** служит для преобразования полярных координат точки в прямоугольные координаты на плоскости (рис. 7.41).

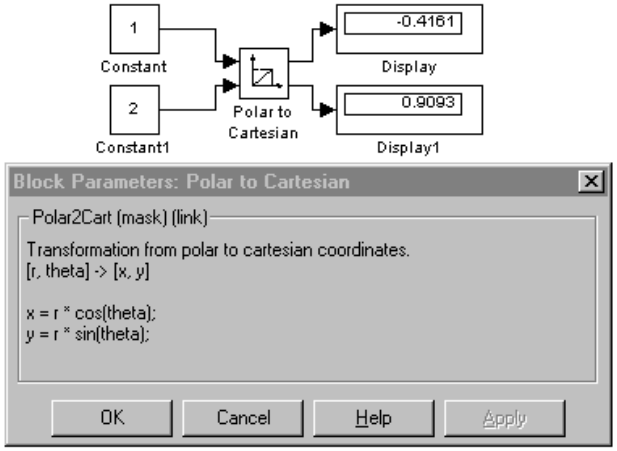


Рис. 7.41. Пример преобразования полярных координат точки на плоскости в прямоугольные координаты

7.4.8. Блок преобразования 3D-координат Cartesian to Spherical

Блок **Cartesian to Spherical** служит для преобразования прямоугольных координат точки в пространстве в сферические координаты (рис. 7.42).

7.4.9. Блок преобразования 3D-координат Spherical to Cartesian

Блок **Spherical to Cartesian** служит для преобразования сферических координат точки пространства в прямоугольные координаты (рис. 7.43).

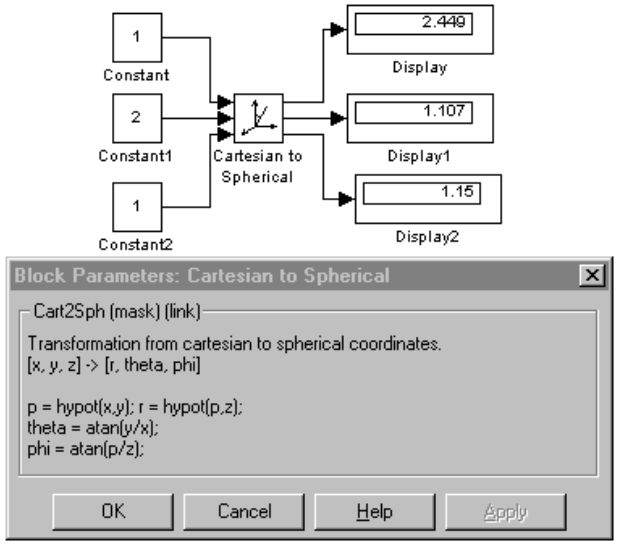


Рис. 7.42. Пример преобразования прямоугольных координат точки в пространстве в сферические координаты

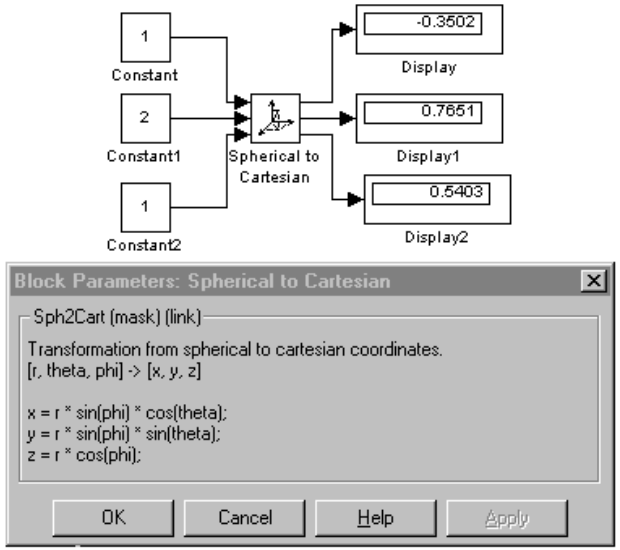


Рис. 7.43. Пример преобразования сферических координат точки в пространстве в прямоугольные координаты

7.5. Библиотека верификации модели – Model Verification

7.5.1. Обзор блоков библиотеки Model Verification

Новой и довольно обширной библиотекой в Simulink 5/6 стала библиотека **Model Verification** – рис. 7.44. По существу, эта библиотека содержит достаточно представительный набор блоков для контроля сигналов моделей.

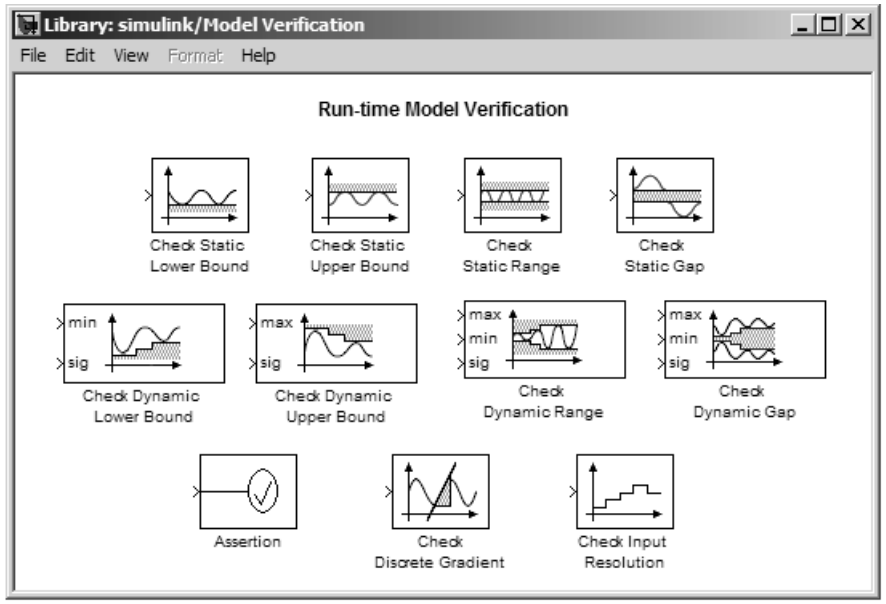


*Рис. 7.44. Окно библиотеки **Model Verification***

Всего данная библиотека содержит 10 блоков. Какого-либо деления на категории или классы библиотека не содержит. Однако нетрудно заметить, что блоки делятся на блоки контроля со статическими уровнями и блоки контроля с динамическими уровнями.

7.5.2. Блоки контроля со статическими уровнями

Блоки контроля со статическими уровнями имеют в названии слово Static. Имеются четыре таких блока:

- контроля минимального статического уровня сигнала – `Check Static Lower Bound`;
- контроля максимального статического уровня сигнала – `Check Static Upper Bound`;
- контроля нахождения сигнала в статической зоне – `Check Static Range`;
- контроля нахождения сигнала вне статической зоны – `Check Static Gap`.

Рассмотрим для примера работу блока `Check Static Lower Bound`. Она показана на рис. 7.45. На вход блока подан синусоидальный сигнал. Блок настроен на порог –0,5, и когда сигнал на входе блока достигает этого уровня, то моделирование прекращается. При этом выдается сигнал ошибки, и построение синусоиды на входе контрольного осциллографа прекращается. Блок приобретает характерную рамку, используемую для индикации ошибки наряду с выдачей окна диагностики.

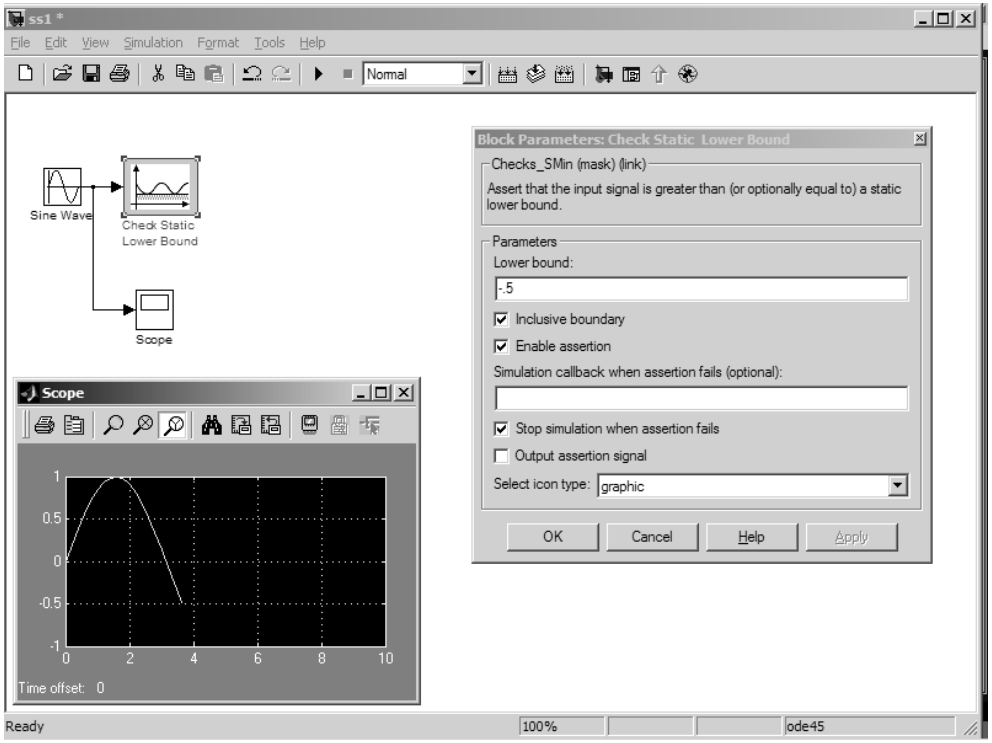


Рис. 7.45. Пример применения блока ***Check Static Lower Bound***

Окно диагностики показано на рис. 7.46. Окно дает информацию о том, в каком блоке наступило прерывание (ошибка), и дает короткий отчет об этом. Команда `Remote highlighting` в позиции `View` меню основного окна системы Simulink позволяет сбросить индикацию ошибки и повторить моделирование.

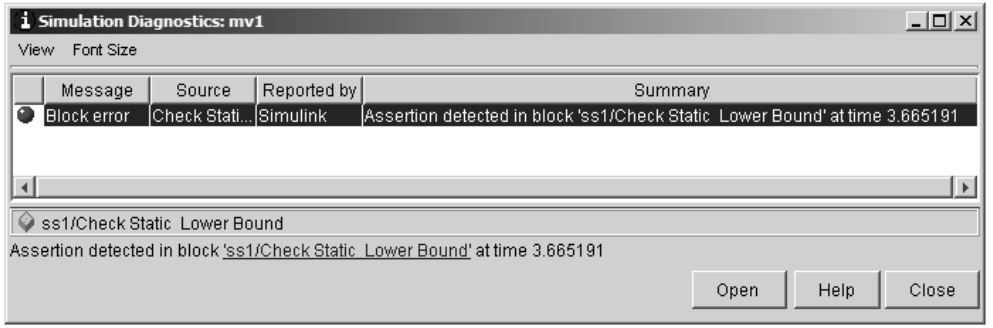


Рис. 7.46. Окно диагностики при прерывании моделирования

Окно параметров блока (рис. 7.46) позволяет задать следующие параметры:

- `Lower bound` – контролируемое нижнее граничное значение сигнала;
- `Inclusive boundary` – включить граничное значение сигнала в условие проверки;
- `Enable Assertion` – включить реакцию (при отключении этой опции проверка осуществляется при отсутствии реакции);
- `Simulation callback when assertion fails` – подключение М-функции, которая будет исполняться при наличии реакции;
- `Stop simulation when asserting fails` – остановка моделирования при невыполнении условия;
- `Output Assertion Signal` – создание дополнительного выходного порта, на который выводится логический сигнал (он меняется с TRUE на FALSE при невыполнении условия проверки);
- `Select Icon Type` – выбор из списка вида пиктограммы блока (`graphic` – графический, `text` – текстовый).

Остальные блоки имеют аналогичную идеологию применения и аналогичные параметры. Блоки с зонами контроля имеют два основных параметра для установки верхнего и нижнего уровней зоны.

7.5.3. Блоки динамического контроля

Для контроля уровня сигнала в условиях изменения контролируемого уровня служат четыре блока:

- контроль минимального динамического уровня сигнала – `Check Dynamic Lower Bound`;
- контроль максимального динамического уровня сигнала – `Check Dynamic Upper Bound`;
- контроль нахождения сигнала в динамической зоне – `Check Dynamic`;
- контроль нахождения сигнала вне динамической зоны – `Check Dynamic Gap`.

Для приема изменяющегося уровня первые два блока имеют дополнительный вход, а вторые – два входа (для нижнего и верхнего уровней зоны). Рисунок 7.47

иллюстрирует применение блока `Check Dynamic`, контролирующего вхождение синусоидального сигнала в постепенно сужающуюся зону. Зона задается двумя источниками линейно-изменяющегося сигнала. На рис. 7.47 представлены также окно параметров блока `Check Dynamic` и осциллограммы сигналов – синусоиды и порогов, задающих зону.

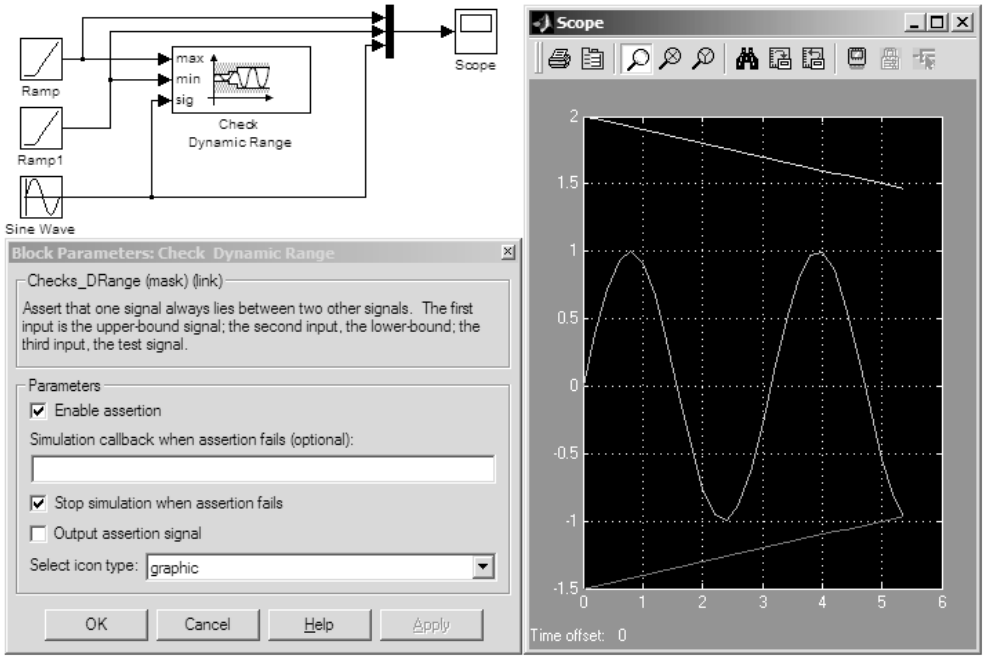


Рис. 7.47. Пример применения блока `Check Dynamic` *для контроля наличия сигнала внутри динамически изменяемой зоны*

Аналогичным образом нетрудно построить модели, демонстрирующие работу других блоков.

7.5.4. Блок контроля нуля Assertion

Блок `Assertion` останавливает вычисления, если будет обнаружено достижение сигналов нулевого уровня. Рисунок 7.48 показывает остановку вычислений в момент, когда ступенчатый сигнал, имеющий значение 1, меняет значение на 0. На рисунке видны закраска изображения блока в момент останова, окно вывода диагностического сообщения, окно параметров блока `Assertion` и осциллограмма начальной части ступени – последующая часть не видна, так как моделирование было остановлено.

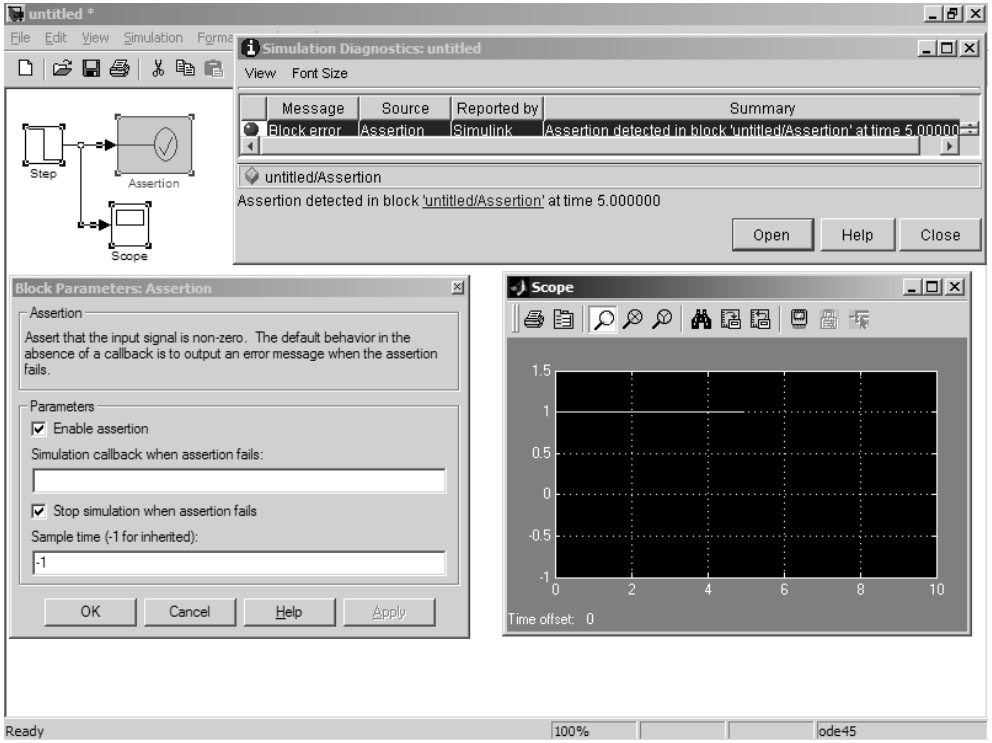


Рис. 7.48. Пример применения блока `Assertion`

Параметры этого блока вполне очевидны и описывались ранее. Следует отметить, что момент достижения нуля должен быть точным. Иначе может произойти пропуск момента пересечения сигналом нуля, например если сигнал синусоидальный.

7.5.5. Блок контроля градиента дискретного сигнала Check Discrete Gradient

Для контроля градиента дискретного сигнала используется блок `Check Discrete Gradient`. Блок останавливает моделирование с выдачей диагностического сообщения, если градиент дискретного сигнала достигает заданной величины. Рисунок 7.49 иллюстрирует применение этого блока. Здесь источник сигнала задает синусоидальный дискретный сигнал с амплитудой 5 и фазовым сдвигом pi/2. Фактически это означает генерацию сигнала косинуса, градиент которого нарастает во времени во время первого четвертьпериода. Остановка моделирования происходит в момент, когда градиент этого сигнала достигает установленного

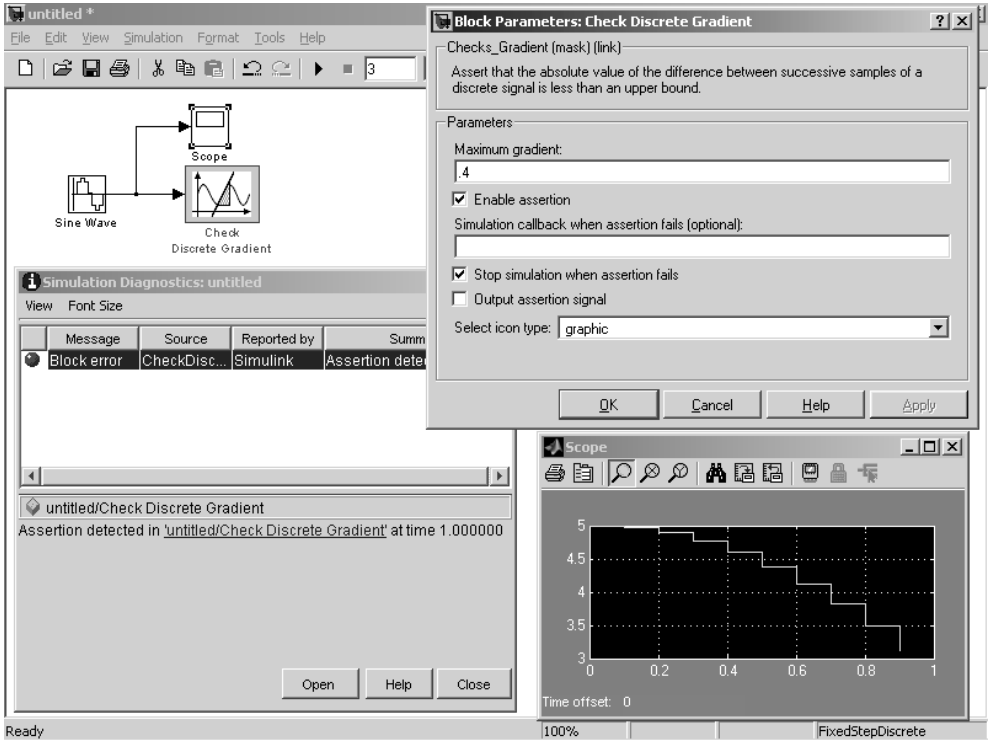


Рис. 7.49. Пример применения блока ***Check Discrete Gradient***

значения 0,4 (см. установки в окне параметров блока). В этот момент обрывается построение осциллограммы дискретного сигнала.

Кроме параметра `Maximum Gradient` (максимальное значение градиента), в окне установки параметров имеется ряд параметров, которые уже были описаны.

7.5.6. Блок контроля разрешения Check Input Resolution

Блок `Check Input Resolution` служит для контроля разрешения, которое задается как $u/10^3$. Если разрешение входного сигнала достигает установленной величины, то моделирование прекращается и выдается диагностическое сообщение – см. пример на рис. 7.50.

В этом примере блок `Step` настроен на генерацию ступеньки с начальным значением 1 и конечным 1,01 с длительностью 4. В момент появления ступеньки блок срабатывает и моделирование прекращается. Параметры в окне установки параметров описаны выше.

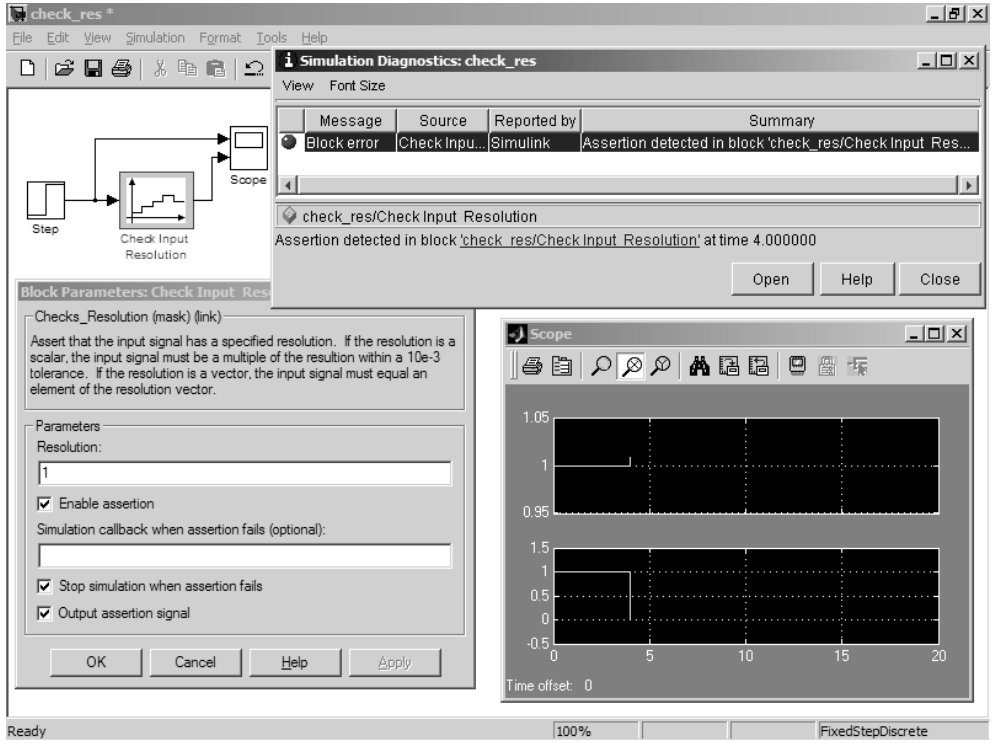


Рис. 7.50. Пример применения блока ***Check Input Resolution***

7.6. Библиотека дополнительных утилит Model-Wide Utilities

7.6.1. Обзор блоков библиотеки Model-Wide Utilities

Новой в Simulink 5/6 библиотекой является **Model-Wide Utilities**. Это небольшая библиотека, окно с блоками которой представлено на рис. 7.51.

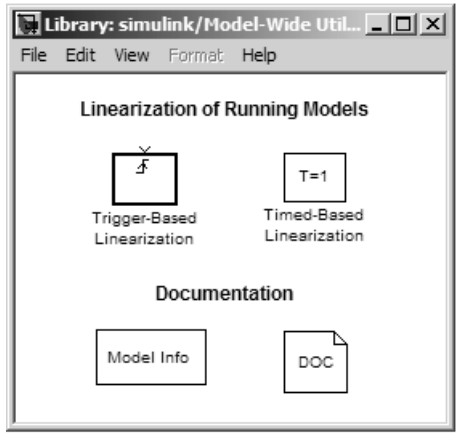


Рис. 7.51. Состав библиотеки ***Model-Wide Utilities***

Библиотека **Model-Wide Utilities** содержит всего четыре блока. Два относятся к линеаризации моделей, а другие два – к сопровождению модели.

7.6.2. Блок линеаризации модели в заданное время Times-Based Linearization

Блок `Times-Based Linearization` служит для линеаризации модели в заданное время. Как уже отмечалось, моделируемая модель имеет представление в пространстве состояний. Она может быть превращена в линейную модель, описываемую матричными уравнениями, в заданный момент времени T. Это и выполняет блок `Times-Based Linearization`.

Заметим, что этот блок не имеет ни входа, ни выхода. Он включается в модель и в заданный момент времени создает в рабочей области системы MATLAB структуру массива с именем *_Timed_Based_Linearization, где * – имя текущей модели. Эта структура имеет следующие поля:

- **a**, **b**, **c**, **d** – матрицы (векторы) линейной модели;
- `State Name` – имена переменных в пространстве состояния модели;
- `Output Name` – имена выходных переменных;
- `Input Name` – имена входных переменных;
- `OpenPoint` – рабочая точка с параметрами переменных состояния x, сигналов u и времени t;
- `Ts` – такт дискретности.

Рисунок 7.52 показывает применение блока `Times-Based Linearization` для простой модели, содержащей источник синусоидального сигнала, интегратор, ограничитель и осциллограф. На рисунке представлены осциллограмма работы модели и окно установки параметров блока `Times-Based Linearization`.

Чтобы увидеть действие блока, надо открыть вьювер рабочей области MATLAB. В правой части рис. 7.52 показано окно системы MATLAB с вкладкой вьювера. В ней видно появление структуры model_lin_Timed_Based_Linearization. Под окном MATLAB показано окно редактора массивов с данными об этой структуре.

7.6.3. Блок линеаризации с запуском Trigger-Based Linearization

Блок `Trigger-Based Linearization` служит для линеаризации модели в момент времени, задаваемый запускающим сигналом, поданным на вход `Trigger`. Рисунок 7.53 поясняет применение этого блока. От примера рис. 7.52 данный пример отличается применением иного блока линеаризации и источника запуска `Step` с параметром T=5.

В окне параметров блока `Trigger-Based Linearization`, помимо обычного параметра `Simple Time`, имеется список режимов запуска `Trigger type`. Возможны следующие режимы запуска:

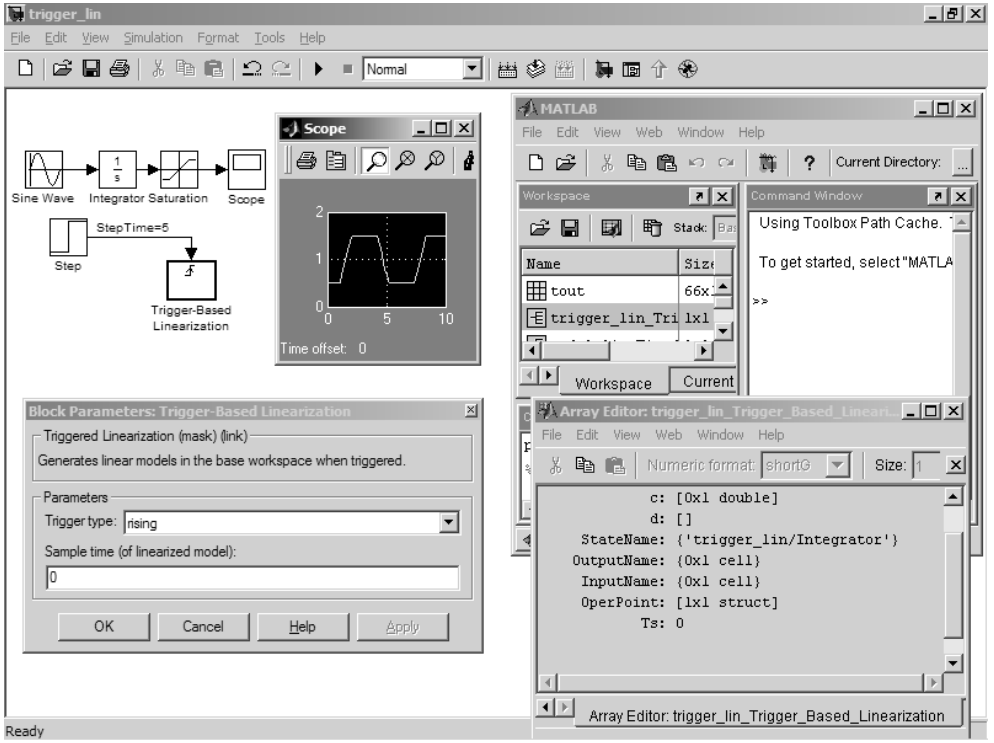


Рис. 7.52. Пример применения блока ***Times-Based Linearization***

- `rising` – по нарастанию (положительному фронту) импульса запуска;
- `falling` – по спаду (отрицательному фронту) импульса запуска;
- `either` – по любому перепаду (фронту);
- `function call` – по S-функции, задающей логику работы модели.

7.6.4. Блок задания информации о модели – Model info

Блок `Model info` служит для задания информации о модели. Этот блок не имеет ни входов, ни выходов и служит для подготовки и вывода (в прямоугольнике из черных линий) информации о модели, которую вводит пользователь. Пример начала работы с блоком `Model info` приведен на рис. 7.54.

Поначалу блок `Model info` пуст. Его активизация мышью вызывает появление окна ввода и редактирования данных о модели. Это окно представлено в правой части рис. 7.54. В левой части окна имеется следующий список данных о модели:

- `Created` – данные о создании модели;
- `Creator` – разработчик модели;

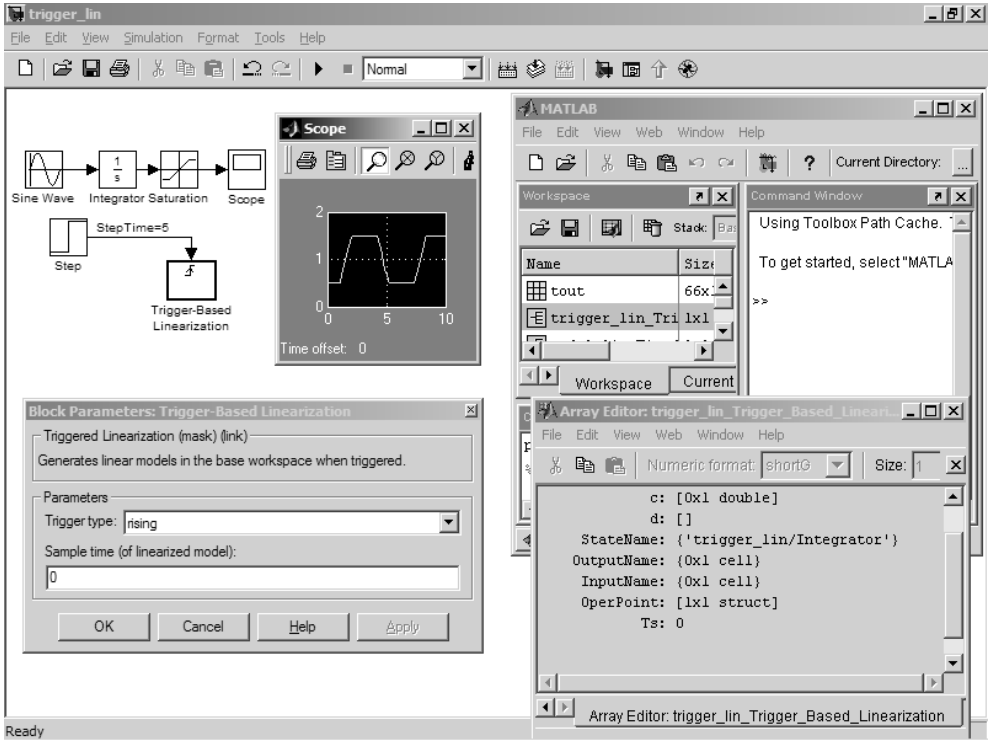


Рис. 7.53. Пример применения блока `Trigger-Based Linearization`

- `Modified by` – изменения модели;
- `Modified data` – дата модификации модели;
- `Modified comment` – комментарий к модели;
- `Model version` – версия модели;
- `Description` – описание модели;
- `Last Modification Date` – последняя дата модификации модели.

Кроме того, в окне можно установить две следующие опции:

- `Horizontal text alignment` – способ выравнивания надписей по горизонтали (`Center` – по центру, `Left` – по левому краю, `Right` – по правому краю);
- `Show block frame` – показ рамки блока.

С помощью кнопки –> соответствующую строку описания можно ввести из левой половины окна в правую и задать соответствующую надпись. После составления описания модели достаточно нажать кнопку **OK**. Вид блока изменится – вместо надписи появятся введенные пользователем данные. Блок можно растянуть, с тем чтобы все введенные данные были видны, – рис. 7.55.

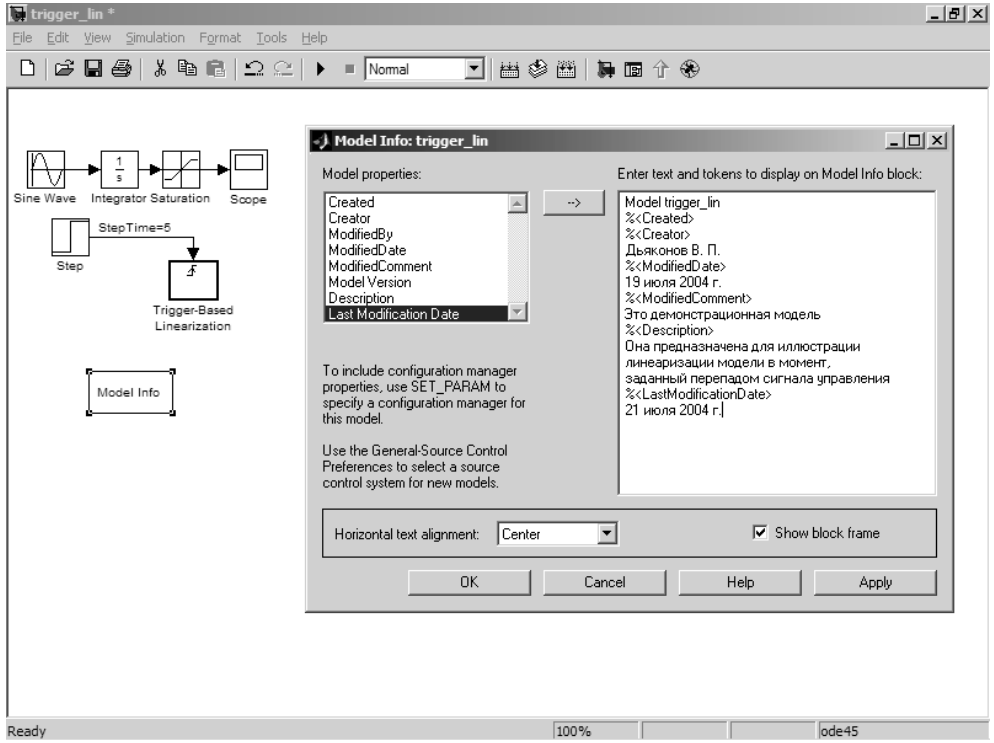


Рис. 7.54. Начало подготовки информационного блока модели

7.6.5. Блок документирования модели – DocBlock

Блок `DocBlock` служит для записи данных о модели с помощью редактора системы MATLAB. Двойной щелчок левой клавишей мыши на этом блоке открывает текстовый редактор (рис. 7.56) и позволяет записать в его окне любой текст. При закрытии окна редактора текст будет сохранен.

7.7. Новые нелинейные и дискретные блоки Simulink 6

7.7.1. Новые нелинейные блоки Simulink 6.6

В Simulink 6.* библиотека нелинейных блоков Discontinuities несколько расширена. Окно ее с набором блоков представлено на рис. 7.57.

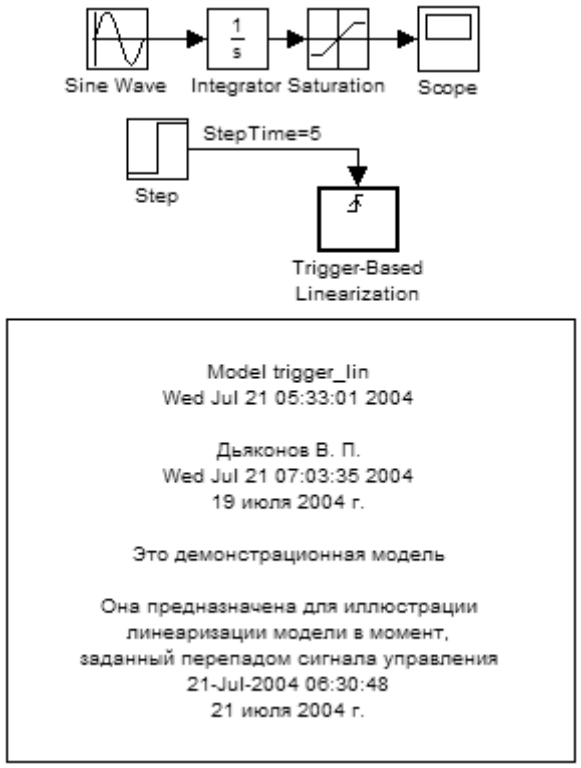


Рис. 7.55. Модель с информационным блоком

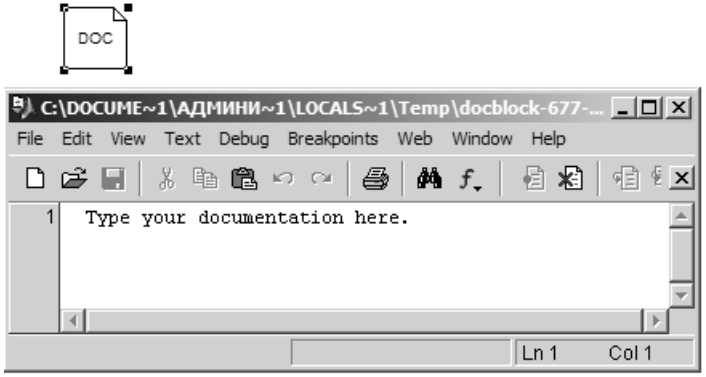


Рис. 7.56. Применение блока `DocBlock`

В состав библиотеки нелинейных блоков вошли три новых блока:

- `Saturation Dynamic` – ограничитель с внешним заданием порогов;
- `Dead Zone Dynamic` – мертвая зона с внешним заданием порогов;

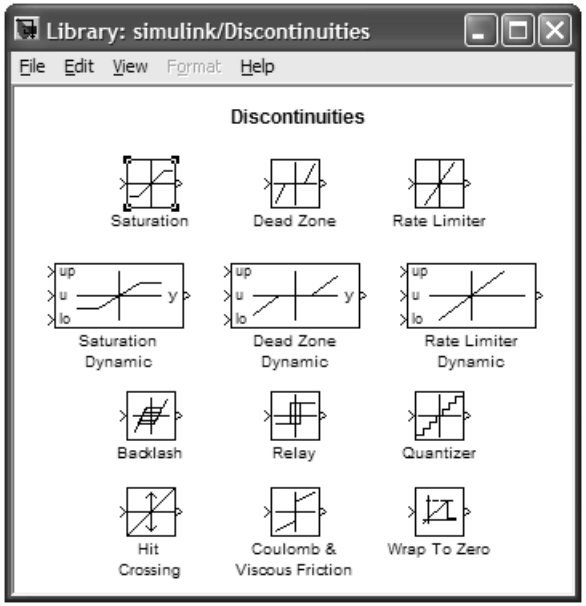


Рис. 7.57. Окно библиотеки нелинейных блоков Simulink 6.6

- `Rate Limited Dynamic` – ограничитель скорости изменения выхода с внешним заданием порогов.

Ранее в состав данной библиотеки входили блоки этого типа с фиксированными порогами, которые задавались в окне установки их параметров и не могли изменяться по ходу моделирования. Новые блоки имеют по три входа, один из которых сигнальный и два других служат для задания верхнего и нижнего порогов. Благодаря этому пороги могут быть динамическими и меняться по мере моделирования.

В состав этой группы блоков введен еще один блок – `Wrap To Zero`. Этот блок имеет единственный параметр – порог `Threshold`. Если входной сигнал превышает порог, уровень выхода становится нулевым, иначе он равен уровню входа.

7.7.2. Новые дискретные блоки Simulink 6.6

Набор дискретных блоков в Simulink 6.* также несколько пополнился. Это видно из сравнения окна библиотеки **Discrete** таких блоков, показанного на рис. 7.58 с окном аналогичной библиотеки Simulink 5.1 (рис. 7.10).

Как видно из сравнения окон, в библиотеку дискретных блоков Simulink 6.6 добавлены следующие блоки:

- **Difference** – дискретная разность;
- **Discrete Derivative** – дискретная производная;
- **Transfer Fcn First Order** – преобразование Fcn первого порядка;

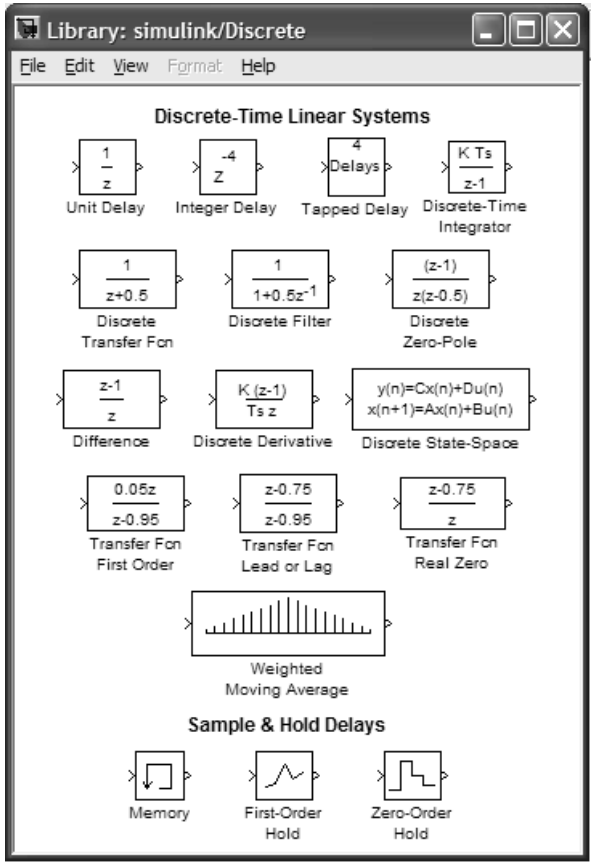


Рис. 7.58. Окно библиотеки дискретных блоков Simulink 6.6

- **Transfer Fcn Lead or Lad** – преобразование Fcn;
- **Transfer Fcn Real Zero** – преобразование Fcn;
- **Weighed Moving Average** – сглаживание методом скользящего среднего.

Урок 8

Подготовка и применение подсистем

Несмотря на обилие блоков в библиотеках пакета расширения Simulink, в практике моделирования всегда могут потребоваться блоки с особыми функциями и возможностями. В Simulink возможно создание таких блоков (субблоков, или подсистем) самим пользователем. Эта возможность описана в данном уроке.

8.1. Общие сведения о подсистемах

Как уже ясно, хотя бы из примеров, рассмотренных выше, 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.

8.2. Создание подсистемы из части основной модели

8.2.1. 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. 8.1.

Запустив модель, можно наблюдать ее работу, что иллюстрируют осциллограммы виртуального осциллографа, подключенного к выходу мультиплексора.

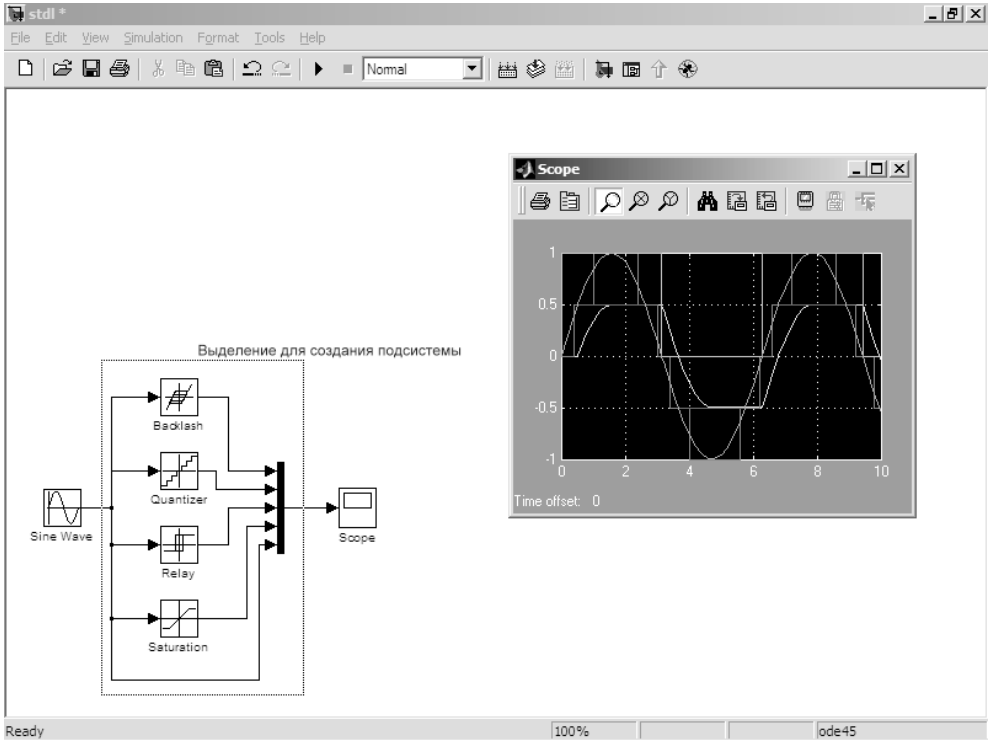


Рис. 8.1. Исходная модель устройства

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.

8.2.2. 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. 8.1).

Отпустив левую кнопку мыши, можно наблюдать, как попавшие в прямоугольник блоки выделяются (рис. 8.2).

Теперь надо выбрать пункт меню **Edit** ⇒ **Create Subsystem** (на рис. 8.2 меню **Edit** раскрыто).

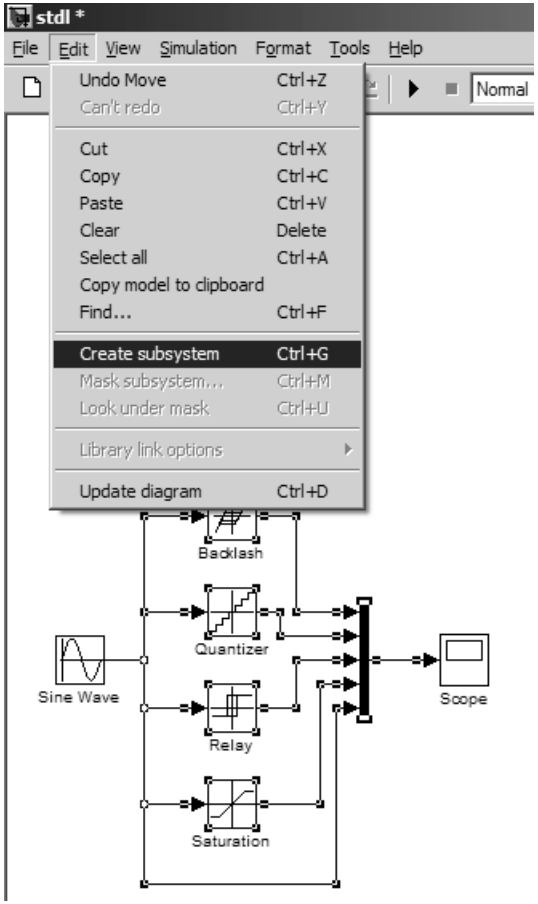


Рис. 8.2. Подготовка к созданию подсистемы

8.2.3. 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. 8.3 показана полученная новая модель с подсистемой после редактирования модели переноса ее мышью в левый верхний угол окна и замены подписей блоков.

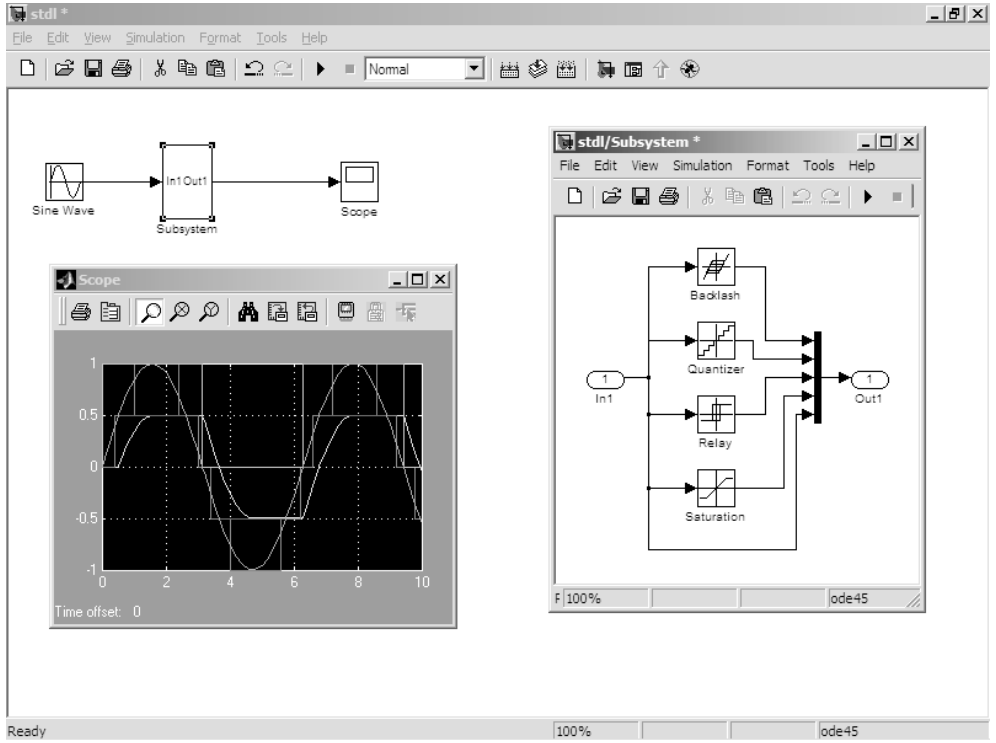


Рис. 8.3. Модель с подсистемой

8.2.4. 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. 8.3 внутри окна модели.

8.2.5. 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 овалами с номером внутри и подписями.

Благодаря этим портам подсистема включается в состав основной модели. Если выделенный под подсистему блок содержит несколько входов и выходов, то

в подсистеме появятся 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.

8.2.6. 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. 8.4).

Браузер удобно использовать для сложных моделей, когда подсистем много и они образуют древовидную структуру. Эту структуру можно наблюдать в левом окне браузера.

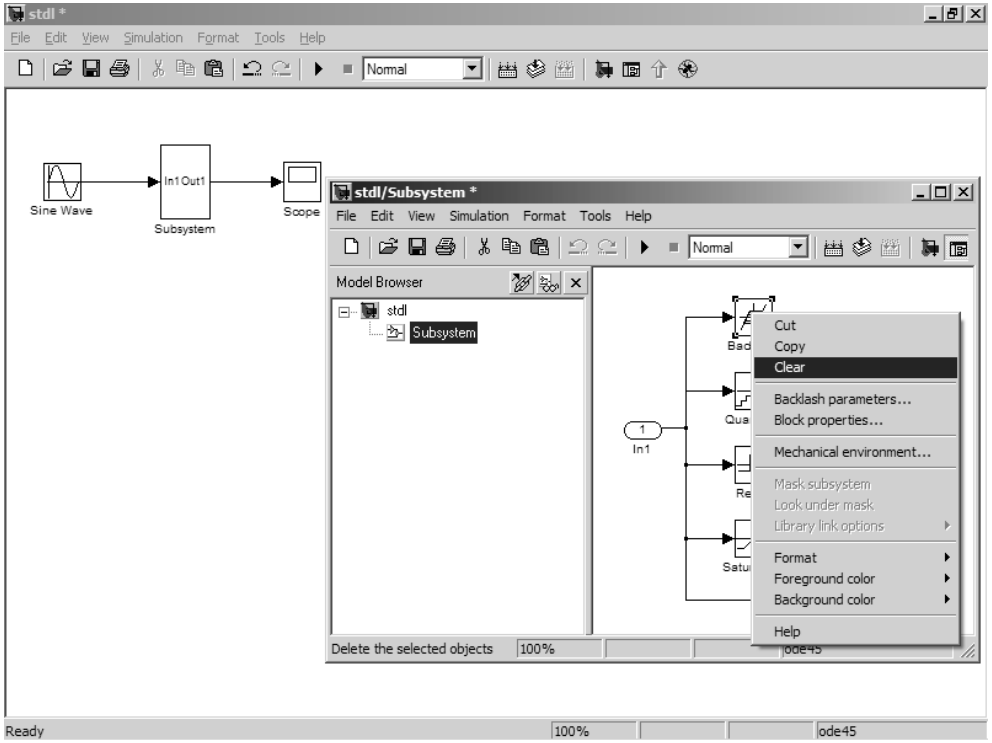


Рис. 8.4. Пример работы с браузером модели

8.2.7. Модификация и редактирование подсистемы

Модификация и редактирование подсистемы ничем не отличаются от этих операций для моделей, описанных ранее. Поэтому ограничимся простым примером – замены в уже созданной подсистеме одного блока на другой.

Подготовка к удалению блока **Backlash** показана на рис. 8.4. 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 `Coulomb & Viscous Friction`. Это и показано на рис. 8.5.

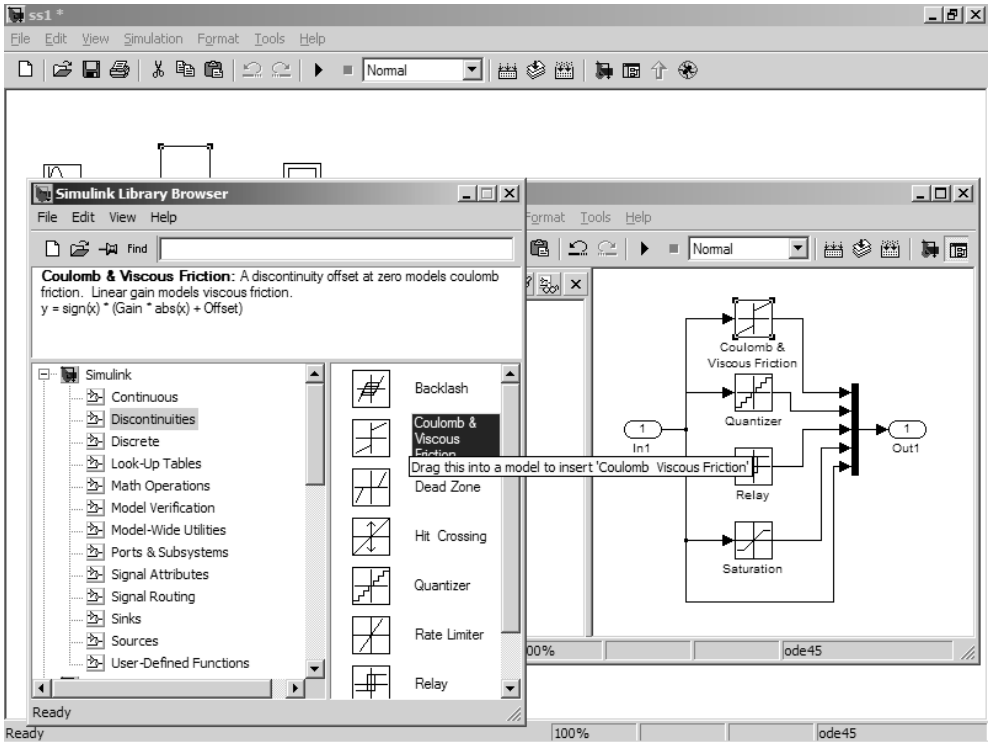


Рис. 8.5. Пример замены одного блока на другой в подмодели

Нажатием кнопки вызова библиотеки в панели инструментов окна подсистемы вызывается окно библиотеки (рис. 8.5). 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 8.6 показывает повторный запуск модели с модифицированной подсистемой.

Таким образом, благодаря применению подсистем можно корректировать последние, не меняя основной модели.

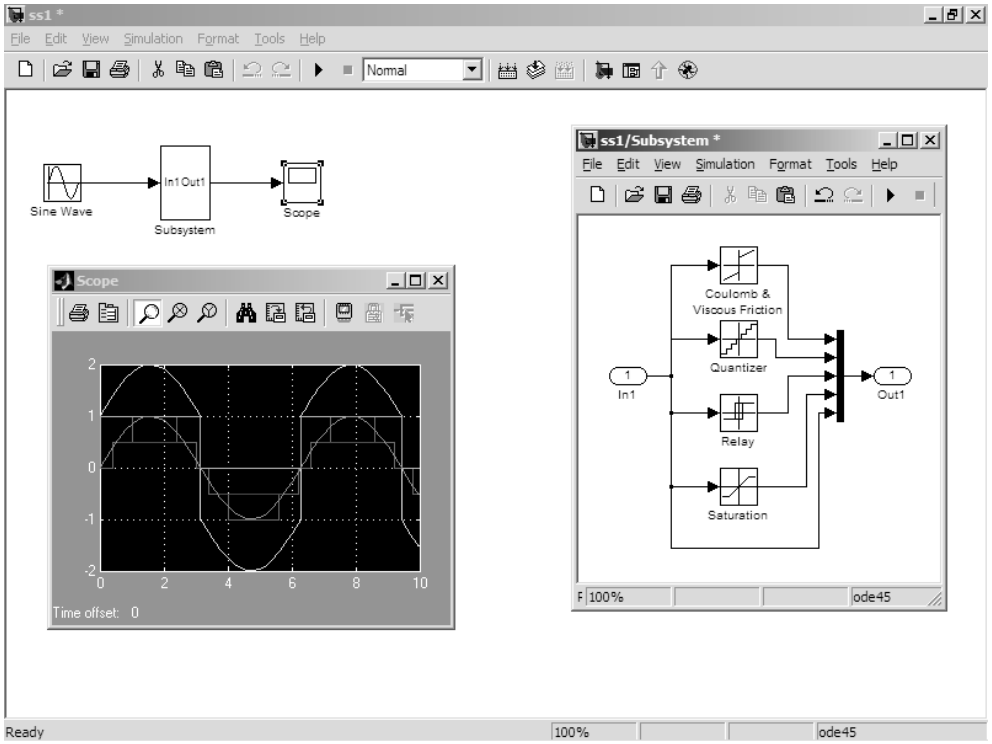


Рис. 8.6. Работа с моделью после коррекции ее подсистемы

8.2.8. Задание свойств подсистемы

Подсистема имеет окно свойств. Его можно вызвать для просмотра и редактирования с помощью команды меню **File ⇒ Model Properties** окна модели или с помощью контекстного меню правой клавиши мыши (рис. 8.7).

Окно свойств подсистемы имеет три вкладки:

- **Model Properties** – свойства подсистемы;
- **Options** – опции, задающие формат свойств;
- **History** – данные об истории подсистемы.

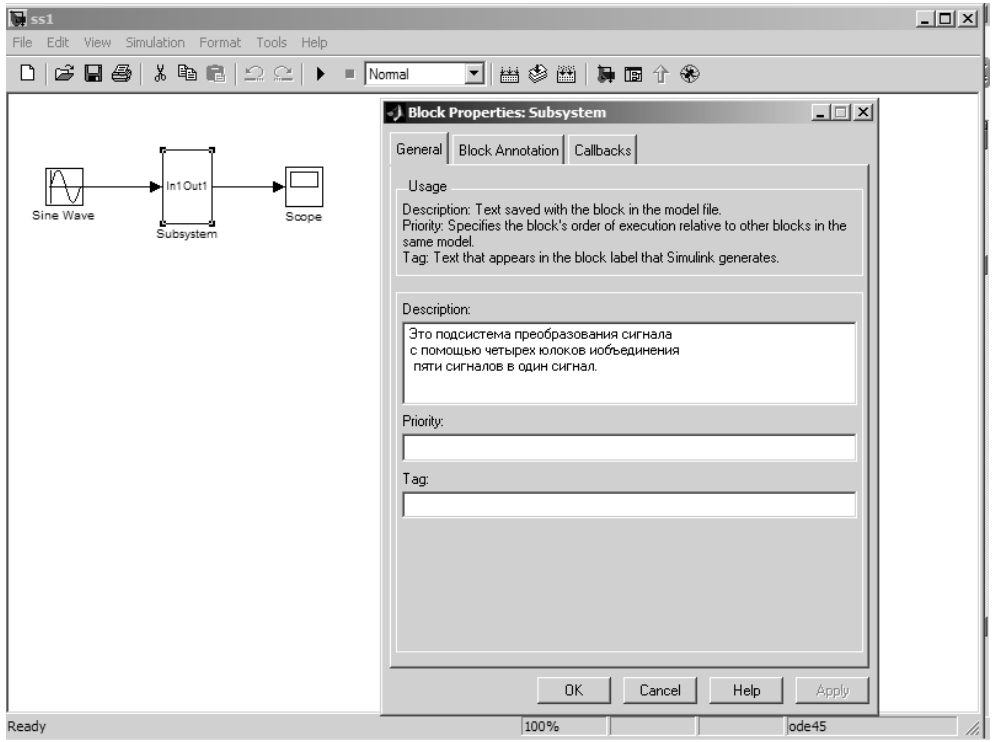


Рис. 8.7. Вызов окна свойств подсистемы

На рис. 8.7 окно свойств подсистемы показано с открытой основной вкладкой. На ней отражаются версия подсистемы и дата ее создания. Другие поля вначале пусты. В поле **Creator** обычно вводятся 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 ознакомиться самостоятельно. Тем более в связи с тем, что они содержат весьма малый объем информации, вполне очевидный из краткого описания их выше.

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.

8.2.9. Параметры портов ввода и вывода

Порты ввода и вывода являются вполне полноценными блоками и имеют свои окна параметров (рис. 8.8).

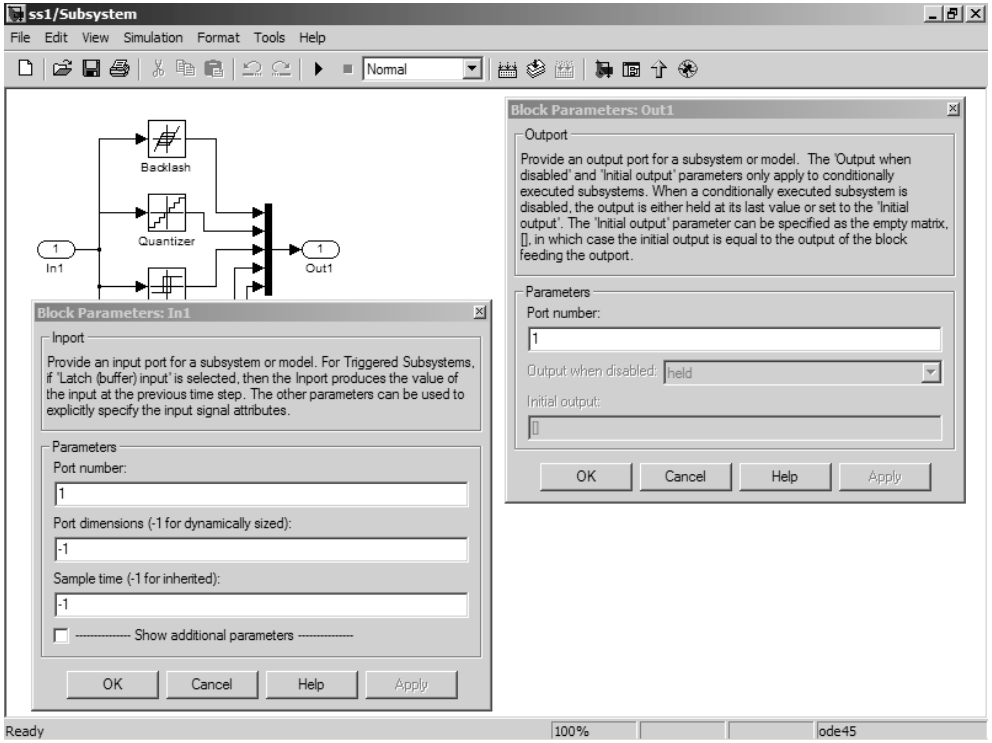


Рис. 8.8. Параметры портов ввода/вывода

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;
- **Initial output** – инициализация выхода.

8.2.10. Обзор библиотеки Ports & SubSystem

Для построения моделей с подсистемами Simulink имеет библиотеку **Ports & SubSystem**, окно которой представлено на рис. 8.9.

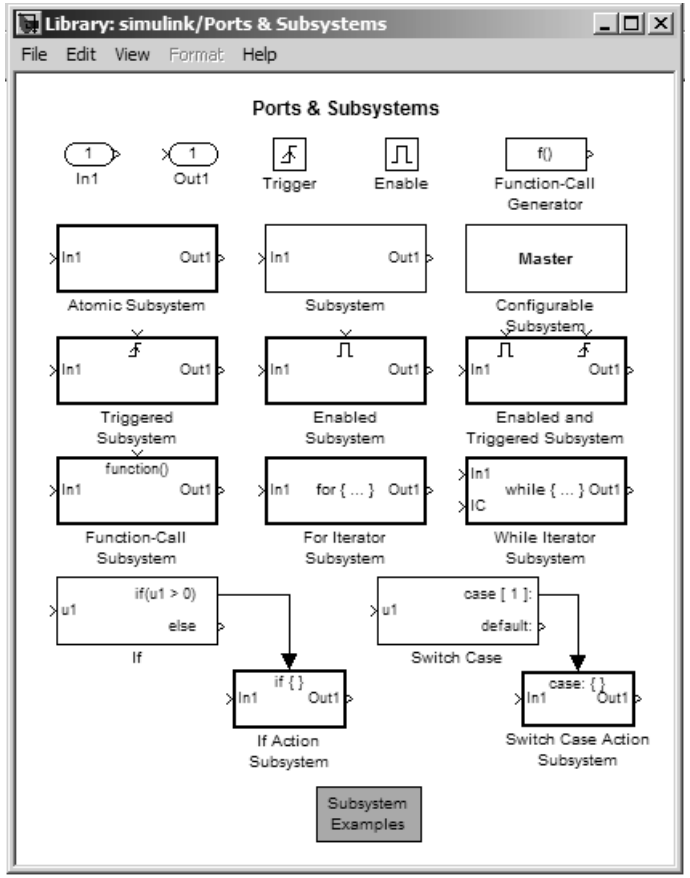


Рис. 8.9. Окно с блоками библиотеки портов и подсистем

Это одна из самых крупных библиотек. Наряду с уже описанными блоками `In*` и `Out*` библиотека содержит блоки организации запуска `Trig`, `Enable` и `Function-cal Generator`, ряд блоков построения различных подсистем и блок демонстрационных примеров.

Из представленных блоков следует особо отметить блок **Atomic Subsystem**. Он служит для создания атомарных (неделимых подсистем). При моделировании блоки такой системы моделируются в первую очередь, а затем уже моделируются другие блоки. Модели, содержащие блоки подсистем Sybsystem, имеют произвольный порядок моделирования блоков.

8.3. Построение подсистем на основе блока SubSystem

8.3.1. Постановка задачи

Предложенный путь создания подсистем путем выделения в их качестве части имеющейся модели не всегда приемлем и не всегда удачен. 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**.

8.3.2. 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**.

8.3.3. Задание подсистемы с помощью блока SubSystem

Прежде всего откроем окно для нашей будущей модели функционального генератора. Далее, открыв окно разделов библиотеки Simulink, перетащим мышью в это окно блок **SubSystem** из раздела **Ports & Subsystems**. Теперь наведем курсор мыши на

этот блок и дважды щелкнем мышью. Откроется окно подсистемы (рис. 8.10). Это окно почти пустое – в нем есть лишь блоки **In1** и **Out1**, соединенные соединением.

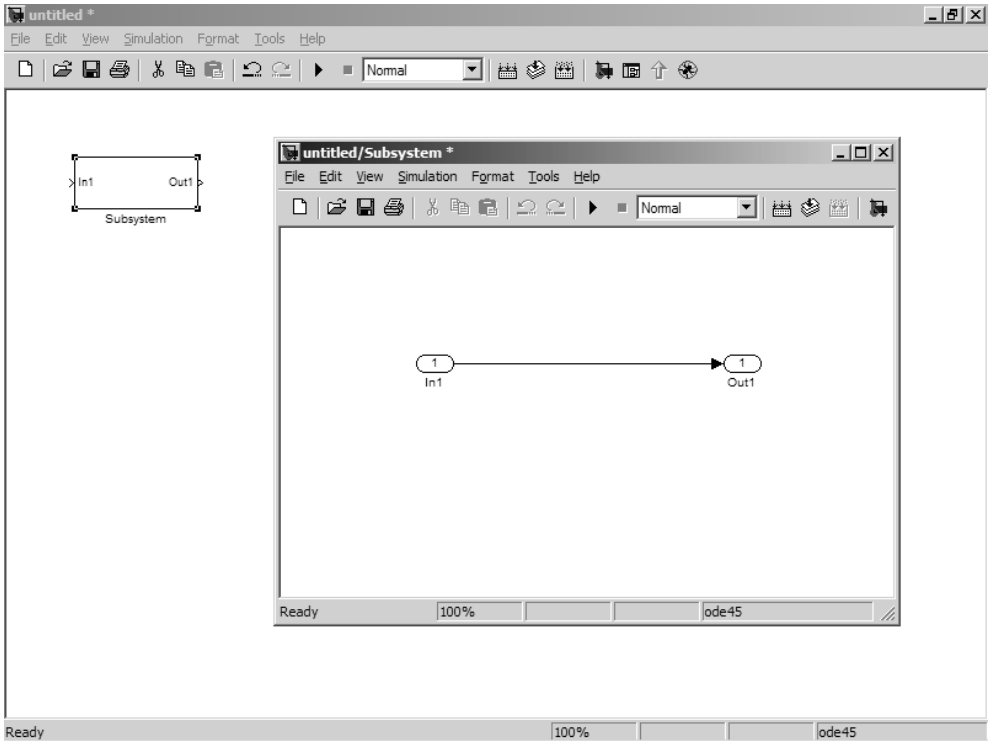


Рис. 8.10. Создание окна подсистемы

Переместив это окно в удобное положение, выделим мышью порт **In1** и соединение, уничтожим их командой **Clear** в позиции меню **Edit** окна подсистемы. Открыв рядом окно разделов библиотеки Simulink, можно начать строить подсистему в соответствии с описанной выше идеей ее реализации. Рисунок 8.11 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. Если подсистема содержит ошибки (а они есть!), то их можно впоследствии исправить.

8.3.4. 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 трехвхо-

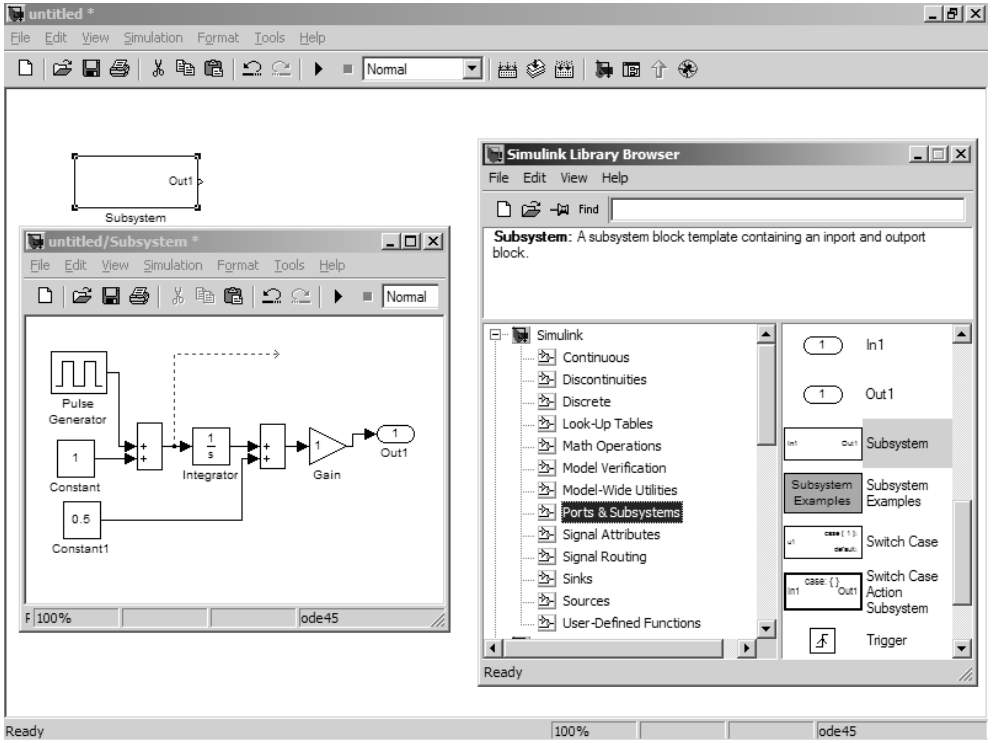


Рис. 8.11. Построение подсистемы

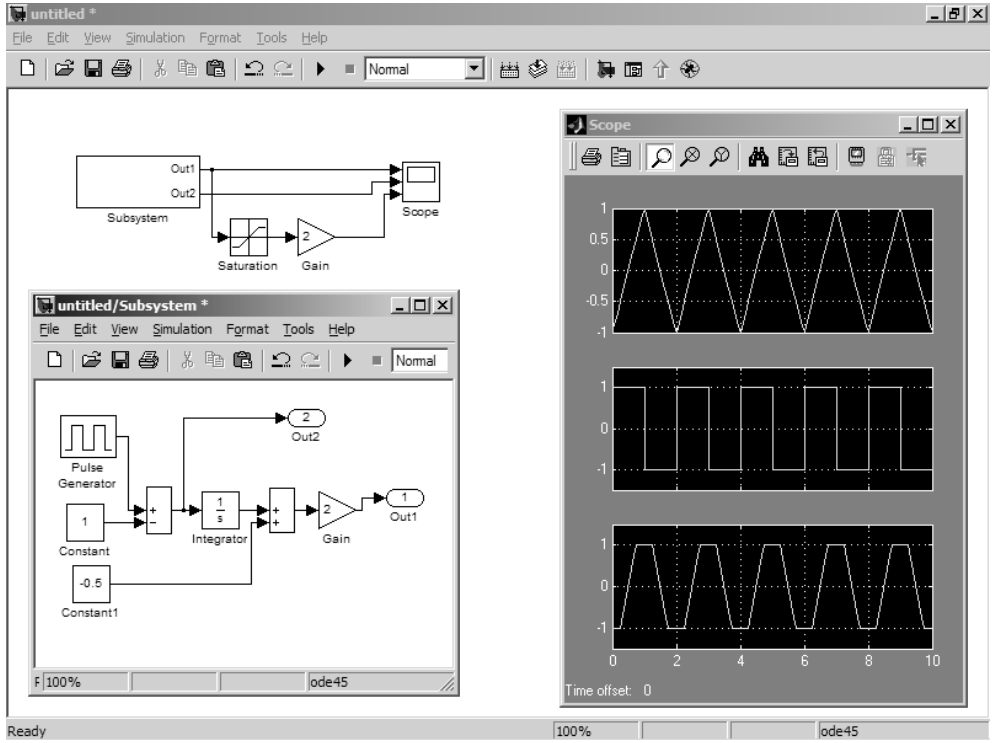


Рис. 8.12. Общая модель функционального генератора

довым 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 8.12 показывает созданную таким образом общую модель.

Остается запустить созданную модель. Перед этим стоит просмотреть еще раз подсистему и устранить обнаруженные в ней ошибки, например сделать нижний вход у первого блока суммы вычитающим, а также задать корректно параметры всех блоков. После этого, пуская моделирование (возможно, несколько раз), надо добиться желаемой работы блоков. Осциллограммы на рис. 8.12 показывают, что мы полностью достигли цели разработки – наша модель формирует все три типа сигнала.

8.4. Управляемые подсистемы

8.4.1. 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 **Port & Subsystems**. 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. 8.13).

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.

*Рис. 8.13. Окно с сообщением о недопустимости переноса блоков **Enable** и **Trigger** в окно основной системы*

8.4.2. Пример создания E-подсистемы функционального генератора

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. 8.14. Расположение блока **Enable** не имеет значения.

Как видно из осциллограмм рис. 8.14, наша подсистема стала управляемой – теперь импульсы на ее выходе формируются только при подаче на управляющий вход перепада сигнала с выхода источника перепада. Желательно, чтобы задержка была кратна периоду генератора прямоугольных импульсов, иначе окажется нарушенной работа интегратора. Следует отметить и 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.

В окне параметров блока `Enable` можно задать следующие параметры:

- `States when enable` – состояние при запуске (выбор из списка: `held` – использовать последнее состояние, `reset` – использовать начальное состояние);
- `Show output port` – показать выходной порт;
- `Enable zero crossing detections` – фиксировать прохождение выходного сигнала через 0.

8.4.3. Создание порта выхода E-подсистемы

Поскольку блок **Enable** всего лишь задает признак управляемости подсистемы, этот блок никуда не подключается. Если в окне параметров блока 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 8.15 иллюстрирует применение дополнительного выходного порта подсистемы.

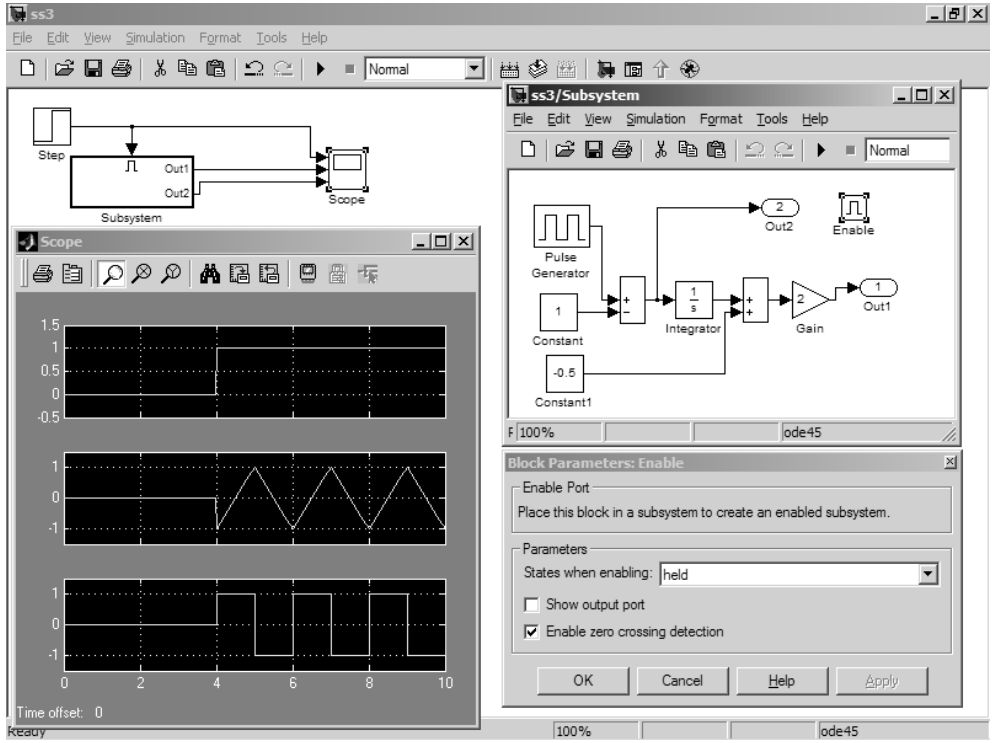


Рис. 8.14. Пример управляемой E-подсистемы

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. 8.15 под подсистемой показано окно параметров ее выходного порта **Out3**.

8.4.4. 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, если управляющий сигнал меняет полярность.

Работа T-подсистем имеет ряд отличительных признаков:

- подсистема работает только на том шаге, на котором произошло изменение полярности управляющего сигнала;

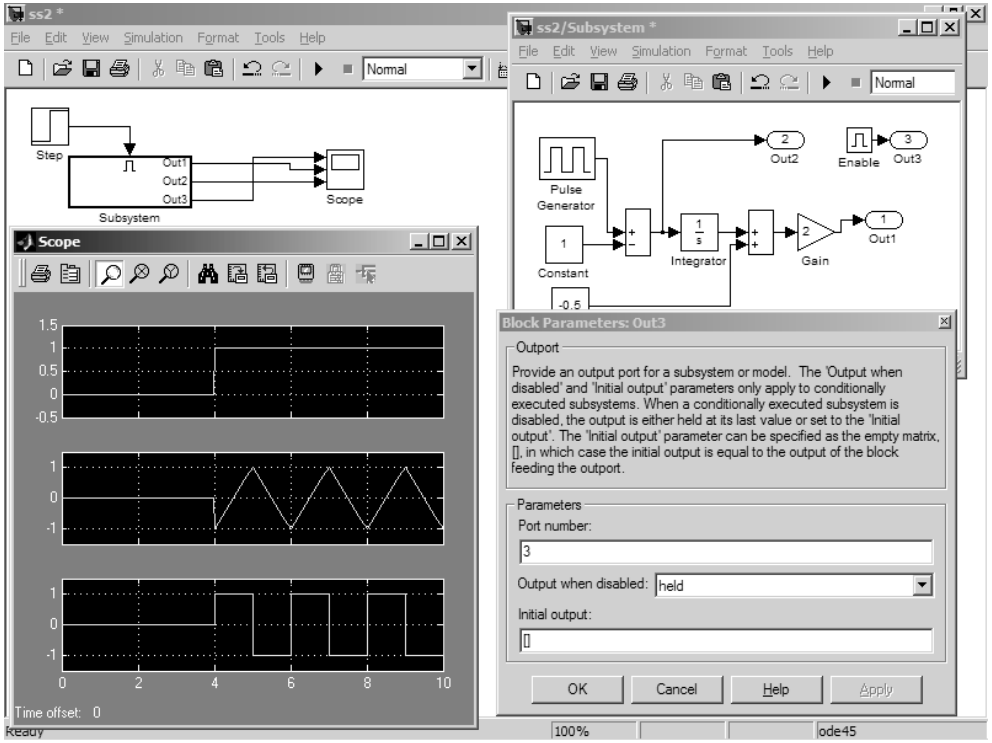


*Рис. 8.15. Пример использования выходного порта блока **Enable** управляемой E-подсистемы*

- 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.

8.4.5. Пример применения T-подсистемы

Прекрасный пример построения T-подсистем имеется среди новых демонстрационных примеров пакета Simulink. Он представлен на рис. 8.16.

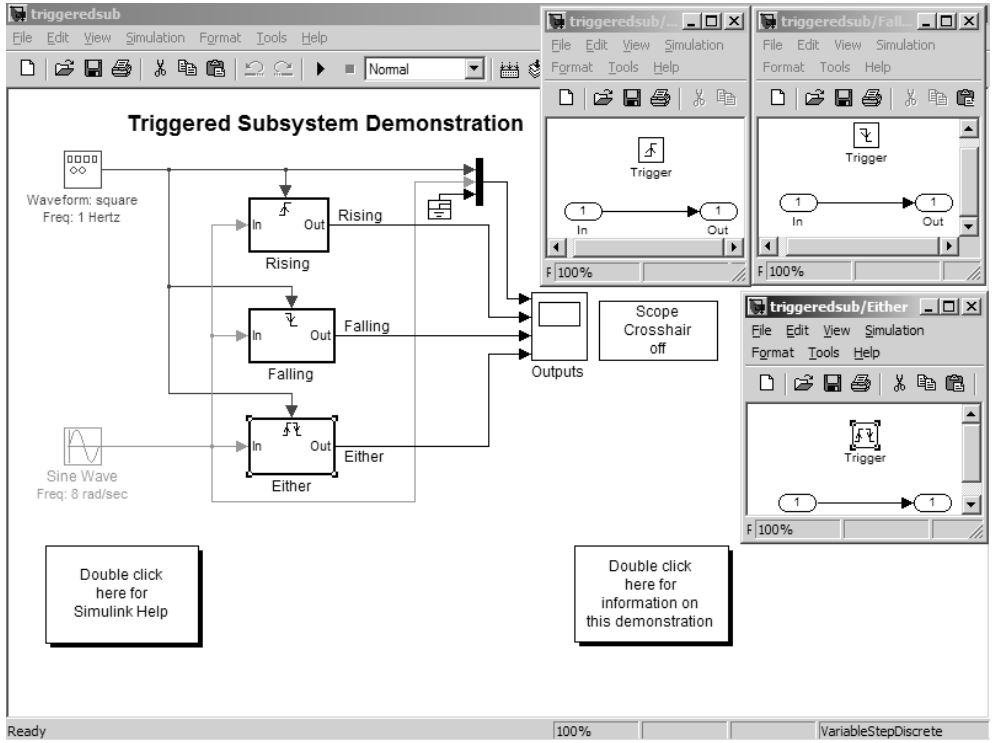


Рис. 8.16. Пример применения простейших T-подсистем

В этом примере заданы три подсистемы-«пустышки». Такие подсистемы создаются блоком **Triggered Subsystem**. Обозначения подобных подсистем и их графическое представление показаны на рис. 8.16. Окно параметров блока **Trigger** одной из подсистем представлено на рис. 8.17.

В окне параметров блока `Trigger` можно задать ряд описанных выше параметров. Основным является параметр `Trigger type` – тип триггера (выбор одного из вариантов из списка: `rising` – запуск по положительному перепаду, `falling` – запуск по отрицательному перепаду, `either` – запуск по любому перепаду и `function-call` – запуск S-функцией, задающей логику системы).

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, только тогда, когда это разрешает управляющий порт (вход). Таким образом, она превращается в управляемый ключ, логика которого определяется параметром **Trigger type**. Они соответствуют трем типам T-подсистем с разными вариантами управления.

Этот пример интересен тем, что в нем используются все три возможных типа сигналов управления с внутренней логикой (кроме управления от S-функции).

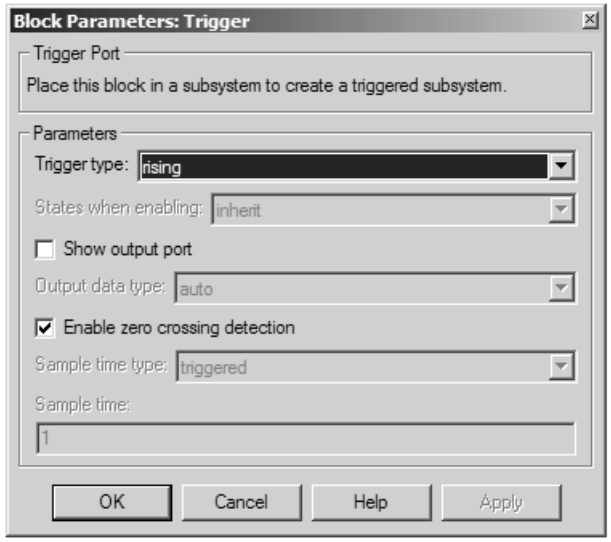


*Рис. 8.17. Окно параметров блока **Trigger***

Создаваемые этой моделью (с ее подсистемами) сигналы представлены на рис. 8.18. Генерация сигналов начинается с начала пуска моделирования. Это важно для понимания отличия Т-подсистем от ЕТ-подсистем, описанных далее.

8.4.6. ЕТ-подсистемы

ЕТ-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 Е-подсистемам, так и Т-подсистемам. Подсистемы этого типа можно вводить с помощью блока или модифицируя блоки подсистем с иной логикой работы.

Для наглядной иллюстрации создания ЕТ-подсистем потребуем от модели рис. 8.17, чтобы первая и третья подсистемы стали ЕТ-подсистемами. Для этого дополним их блоками **Enable**. У этих подсистем появится второй вход, на который можно подать, например, разрешающий сигнал от генератора перепада (рис. 8.19). Таким образом, мы обеспечим генерацию сигналов на выходе первой и третьей подсистем только при наличии сигнала на их дополнительных входах.

Рисунок 8.20 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.

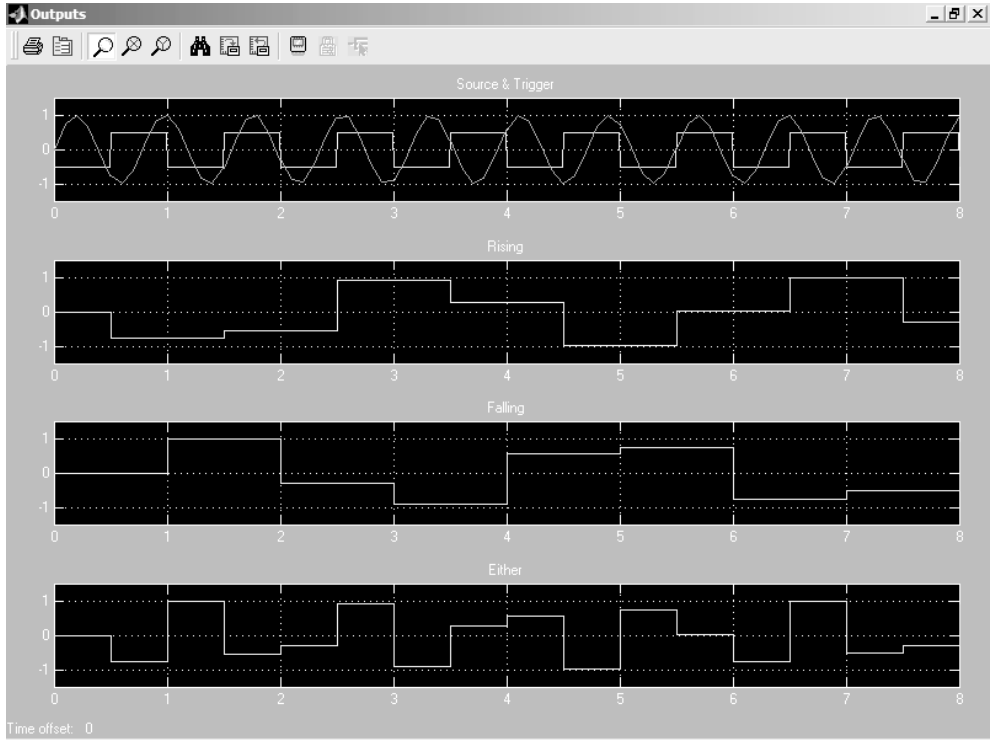


Рис. 8.18. Осциллограммы модели рис. 8.16

8.4.7. 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. 8.19. Здесь блок **Goto** подключен к выходу генератора управляющего перепада, а блоки **From** и **From1** подключены к входам Е-управления первой и третьей подсистем. Таким образом, управляющий перепад подается на эти входы без использования дополнительных соединений.

С позиции применения языков программирования блоки **From** и **Goto** подобны процедурам и обращениям к ним. Такое представление достаточно есте-

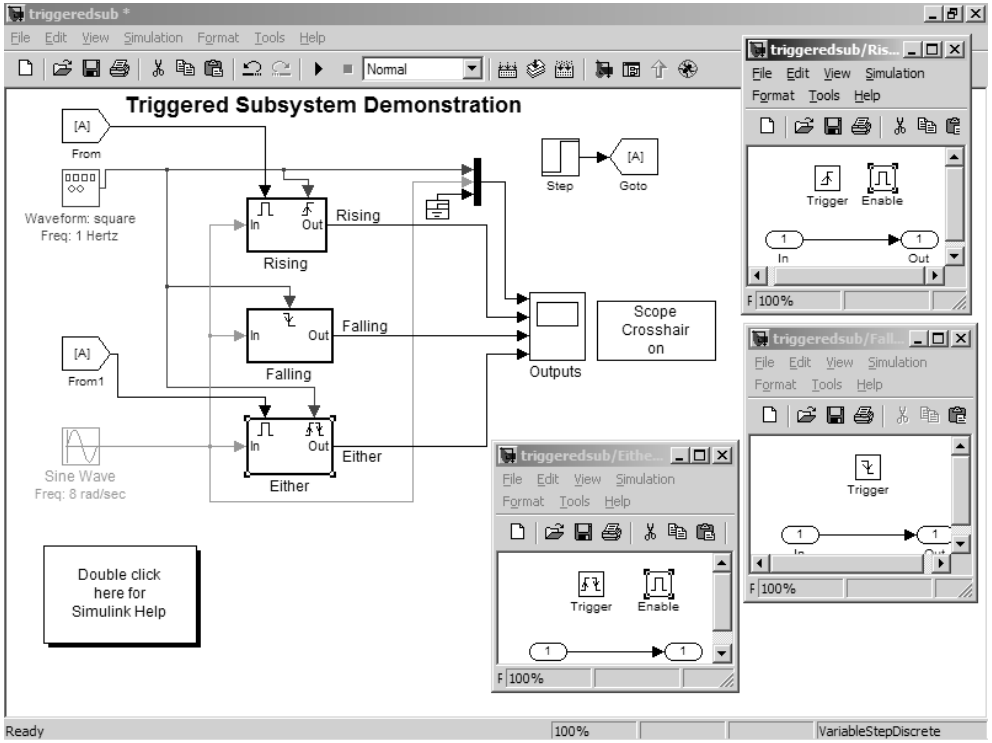


Рис. 8.19. Пример системы с одной T-подсистемой и двумя ET-подсистемами

ственно, если учесть, что Simulink является средством визуального программирования. Кроме того, подобное 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.

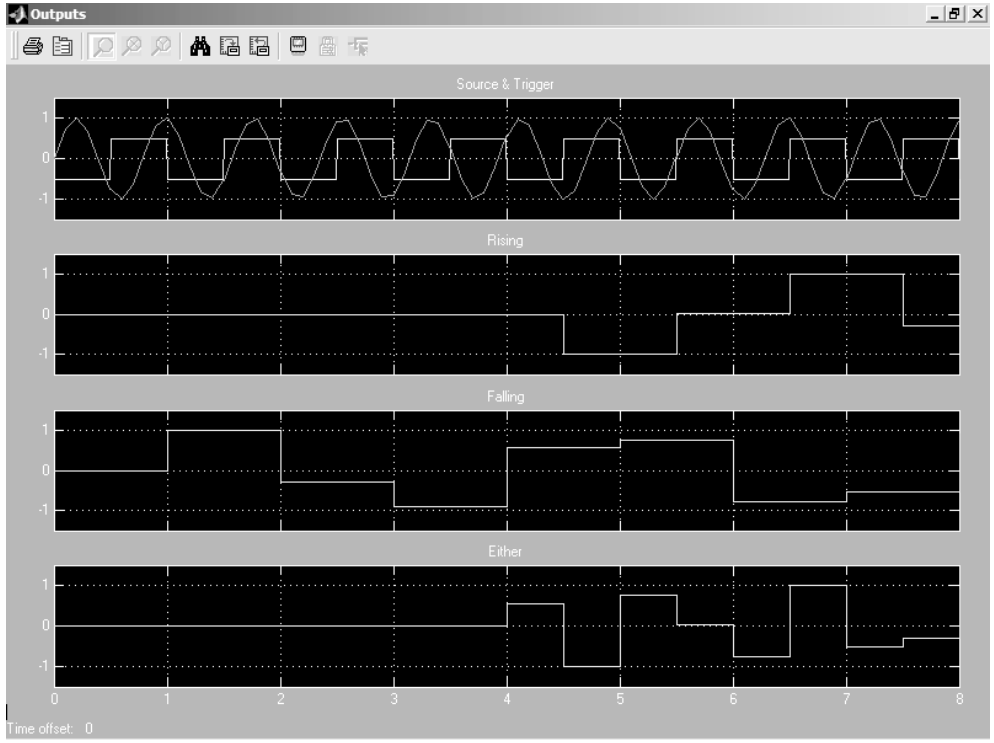


Рис. 8.20. Осциллограммы модели, представленной на рис. 8.19

8.5. Особенности применения подсистем

8.5.1. Виртуальные подсистемы

В Simulink 5 роль подсистем существенно повышена. В отдельный класс выделены так называемые *виртуальные подсистемы*. Построение виртуальных подсистем представляет интерес в основном для опытного пользователя, поскольку такие подсистемы обычно используются как макросы и готовятся на языках программирования типа C или C++. Они поддерживают графическую иерархию в моделях. Ряд таких заранее созданных подсистем включен в новый раздел библиотеки Simulink 5 – Port & Subsystems.

В нем присутствуют следующие виртуальные подсистемы:

- **Configurable Subsystem** – блок конфигурации подсистем;
- **If** – блок задания условных операций;
- **Subsystem** – блок ввода подсистемы в модель;

- **Sybsystem Examples** – открытие окна с примерами описания семантики подсистем;
- **Switch Case** – блок задания оператора переключения.

Нетрудно заметить, что основной блок задания подсистем Subsystem перекочевал из раздела библиотек в новый раздел Simulink 5 – Subsystems. Остальные виртуальные системы по существу являются инструментальными средствами (блоками) для конфигурации подсистем и задания управляющих структур.

8.5.2. Невиртуальные подсистемы

Невиртуальные подсистемы выделяются рамкой из линий двойной толщины (более жирных). Отметим типы встроенных в раздел библиотеки **Port & Subsystems** невиртуальных подсистем. Они перечислены ниже.

Atomic Subsystems – это элементарные (в буквальном переводе атомарные) подсистемы, которые выполняются как единое целое. Любую подмодель с блоками Simulink можно представить в виде подсистемы этого типа.

Enabled subsystems – это E-подсистемы, выполняемые только при наличии на специальном управляющем порте разрешающего сигнала, уровень которого превышает 0.

Triggered subsystems – это триггерные T-подсистемы, управляемые по триггерному управляющему порту и работающие по принципу «спускового крючка». Эти подсистемы нередко используются с расширением StateFlow, обеспечивающим ситуационное моделирование, тогда как расширение Simulink обычно выполняет имитационное моделирование. В блоках таких подсистем нужно задавать эталонное время равным 1.

Function-call subsystems – подсистемы запроса функции. Эти подсистемы используются как m-файлы, за исключением того, что готовятся как блоки Simulink. Данные подсистемы также часто используются с расширением StateFlow и требуют установки эталонного времени блоков равным 1.

Enabled with trigger subsystems – это ET-подсистемы, имеющие свойства E- и T-подсистем.

Action subsystems – это подсистемы, действие которых задается операцией пересечения для подсистем класса Enabled и Function-call subsystems. Они управляются инициатором подсистем, в качестве которого может выступать программный блок условного выражения **If** или переключателя **Switch Case**.

While-subsystems – подсистемы, выполняющие итерационные действия под управлением цикла типа **while (пока)**.

For-subsystems – подсистемы, выполняющие итерационные действия под управлением цикла типа **for** с заданным числом повторения итераций.

8.5.3. Семантика подсистем

Пользователь, желающий детально ознакомиться с описанием (семантикой) подсистем, может активизировать блок **Sybsystem Example** в разделе библиотеки Simulink 5 – **Subsystems** (см. рис. 8.21). Это приводит к выводу окна с описанием подсистем.

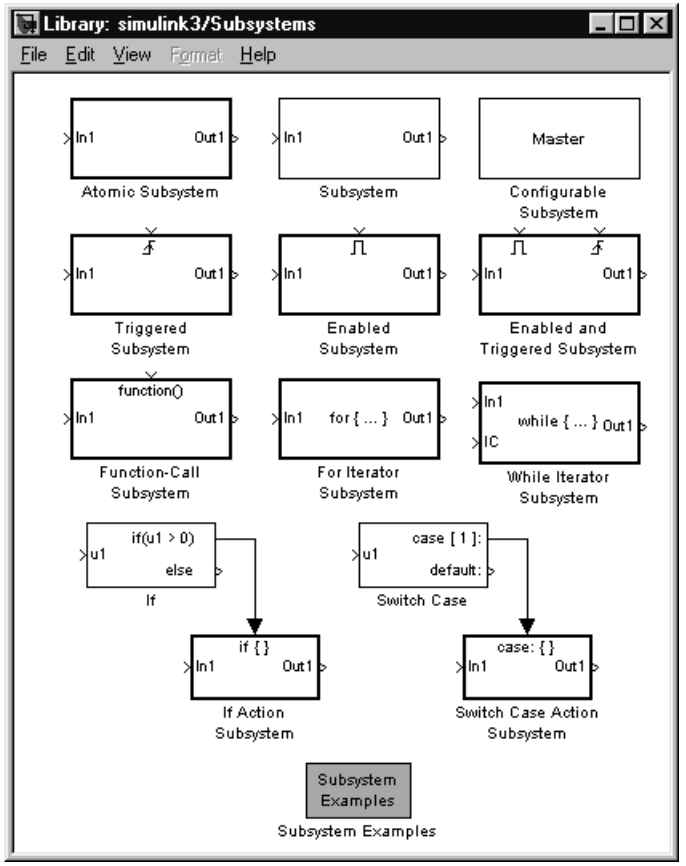


Рис. 8.21. Окно Simulink 3/4 с описанием подсистем

В Simulink 5/6 окно с примерами и описанием подсистем имеет несколько иной вид. Оно показано на рис. 8.22.

Активизировав, в свою очередь, пиктограмму одного из примеров, можно познакомиться с его содержанием. На рис. 8.23 представлено описание отличий виртуальных и невиртуальных подсистем с примерами вывода их моделей.

Разумеется, представленное описание дано на английском языке.

8.5.4. Демонстрационные примеры применения подсистем

Для детального разбора работы подсистем в Simulink 5/6 введено множество новых интересных демонстрационных примеров. Приводить их все в данной книге нецелесообразно, поскольку без компьютера наглядность воспроизведения при-

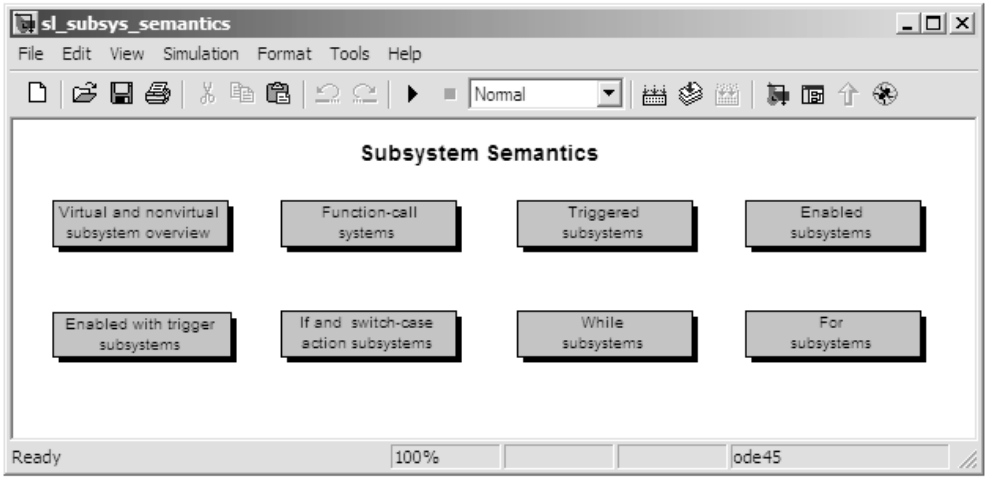


Рис. 8.22. Окно с описанием подсистем

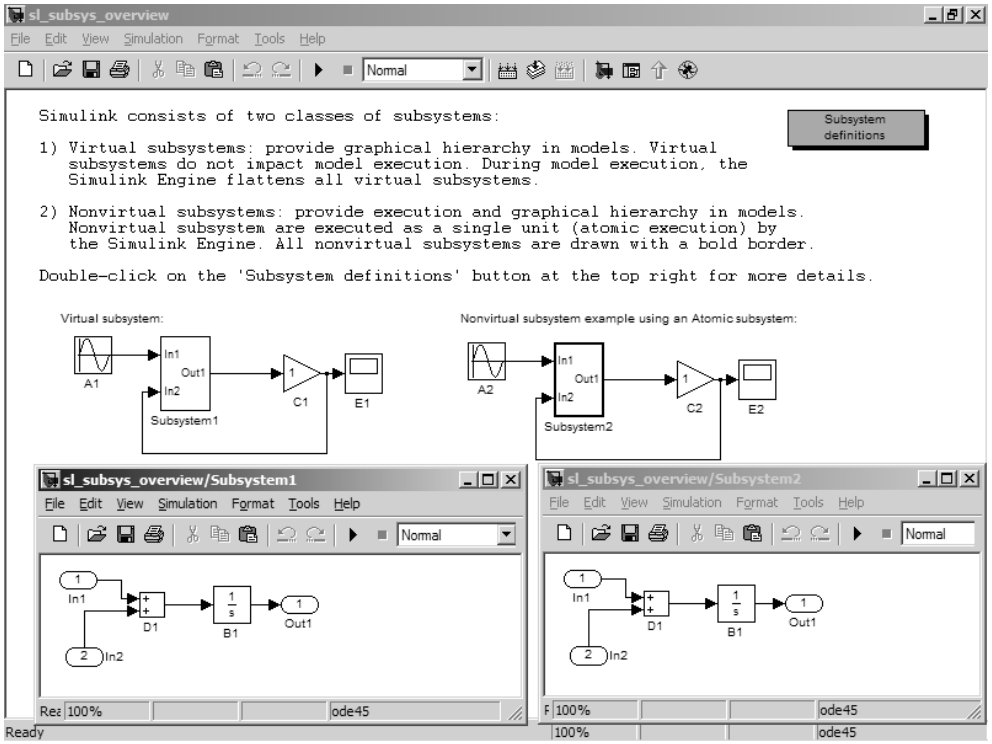


Рис. 8.23. Окно с описанием подсистем

меров низка. А при наличии компьютера пользователь, заинтересованный в детальном знакомстве с техникой применения подсистем, может просмотреть нужные ему примеры самостоятельно. Поэтому будет достаточно описания идеологии организации демонстрационных примеров и доступа к ним. Ограничимся при этом парой наглядных примеров.

Активизируя пиктограмму **Function-call System**, можно вывести окно описаний и демонстрационных примеров для подсистем запроса функций – рис. 8.24.

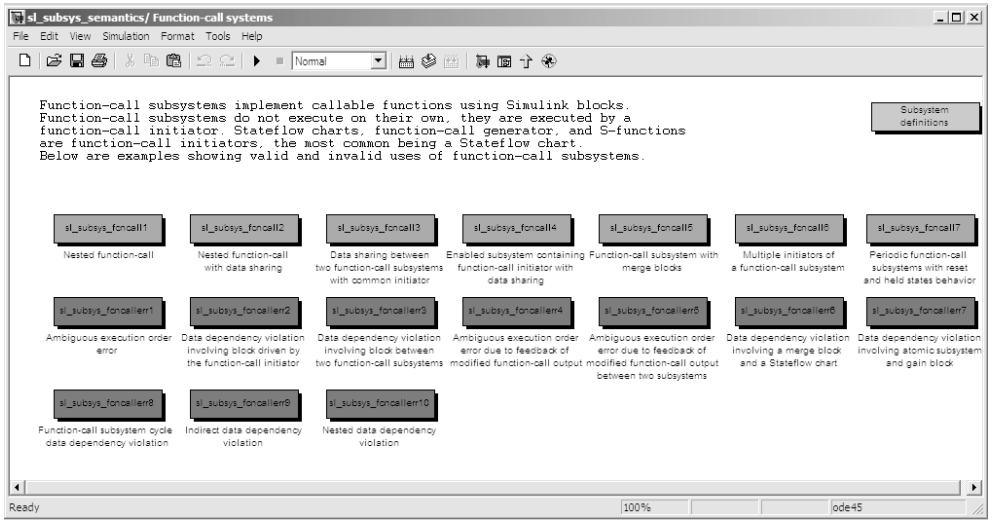


Рис. 8.24. Окно описаний и демонстрационных примеров для подсистем запроса функций

В этом окне зеленые пиктограммы (первый ряд расположен сверху) открывают окна демонстрационных примеров, а красные (два нижних ряда) – окна с описаниями различных вариантов организации подсистем. В качестве первого примера рассмотрим описание первой подсистемы **sl_subsys_fcncallerr1**. Окно с описанием этой подсистемы и ее графической моделью представлено на рис. 8.25.

В принципе, возможен запуск этой подмодели прямо из окна рис. 8.25. Однако такой запуск будет некорректным, поскольку в описании подмодель представлена не полностью. Поэтому воздержимся от запуска, но обратим внимание на то, что в этой модели представлены действующими все связи между блоками. Из описания сверху модели следует, что ее действие заключается в автоматическом прерывании входных связей, относящихся к блокам f и g, – соответственно f() и g().

Теперь активизируем пиктограмму первого демонстрационного примера. Появится окно с ним, в котором будет представлена та же подмодель, что и на рис. 8.25, хотя и с несколько иным текстовым оформлением. Если пустить данную модель на исполнение, то можно увидеть удаление указанных выше связей. Так что окно

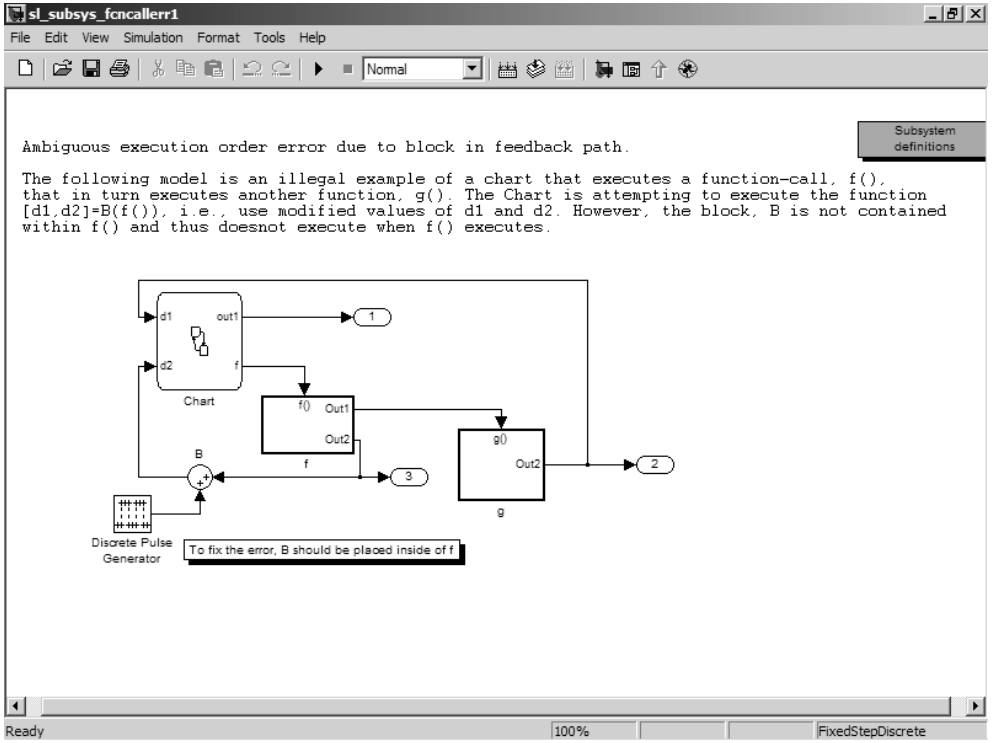


*Рис. 8.25. Окно с описанием подсистемы **sl_subsys_fcncallerr1***

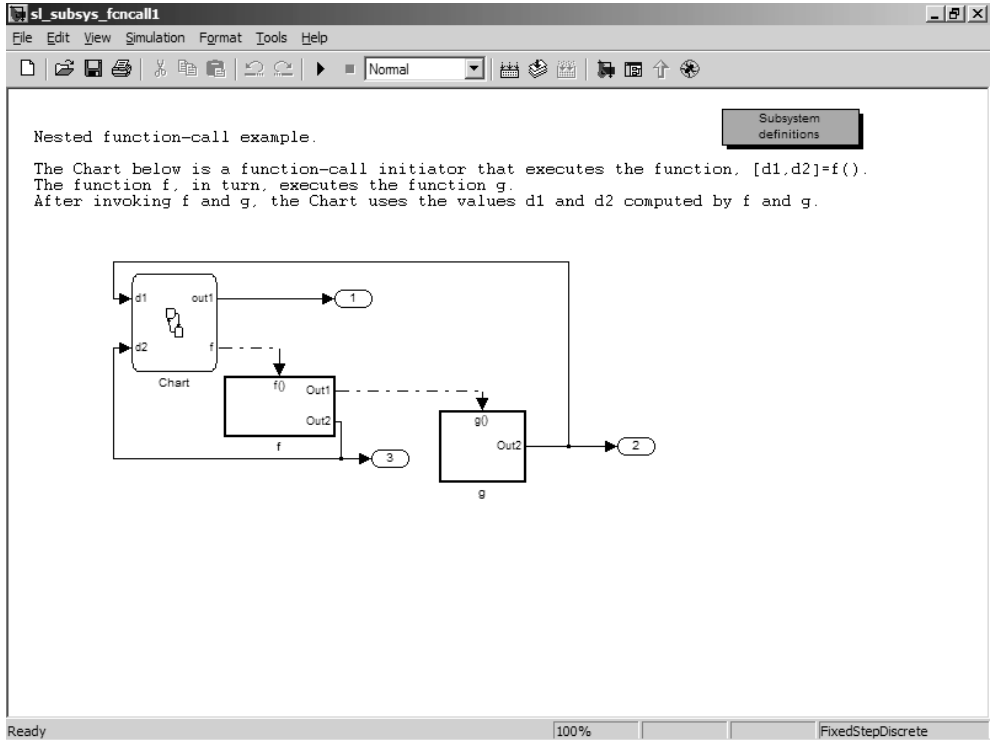


*Рис. 8.26. Окно с примером применения подсистемы **sl_subsys_fcncallerr1** после запуска примера*

примера после запуска примет вид, представленный на рис. 8.26. В нем удаленные связи показаны пунктиром, тогда как сохраненные связи даны сплошными линиями. В данном случае в этом и заключается работа модели.

Теперь рассмотрим еще один из двух примеров на работу подмодели типа **For-subsystem**. Вид его окна после запуска представлен на рис. 8.27. В данном случае в организованном цикле типа **for** (его программная запись представлена под моделью) в цикле вычисляются значения тригонометрической функции синуса, и они выводятся в полях вывода регистратора. Под подмодельюю представлена подмодель вычисления синуса второго уровня вложенности. Вычисления выполняются для вектора значений аргумента синуса от 0 до 1 с шагом 0,2 (всего 6 значений).

Уже из приведенных примеров вполне очевидно, что техника задания подмоделей в Simulink 5 поднялась на новый уровень задания и моделирования структур с переменными связями (и, следовательно, с переменной структурой), а также со структурной реализацией типовых управляющих структур. Они подобны структурам, используемым в языке программирования системы MATLAB (см. урок 10 с подробным описанием сути действия управляющих структур).

8.6. Маскированные подсистемы

8.6.1. 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 выше, такими качествами не обладают. Главное отличие подсистем от блоков – в том, что подсистемы не имеют своей уникальной пиктограммы и 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, порой достаточно сложную структуру. В результате такая подсистема в деталях не видна и воспринимается как библиотечный модуль. Маскированные подсистемы обладают рядом важных достоинств:

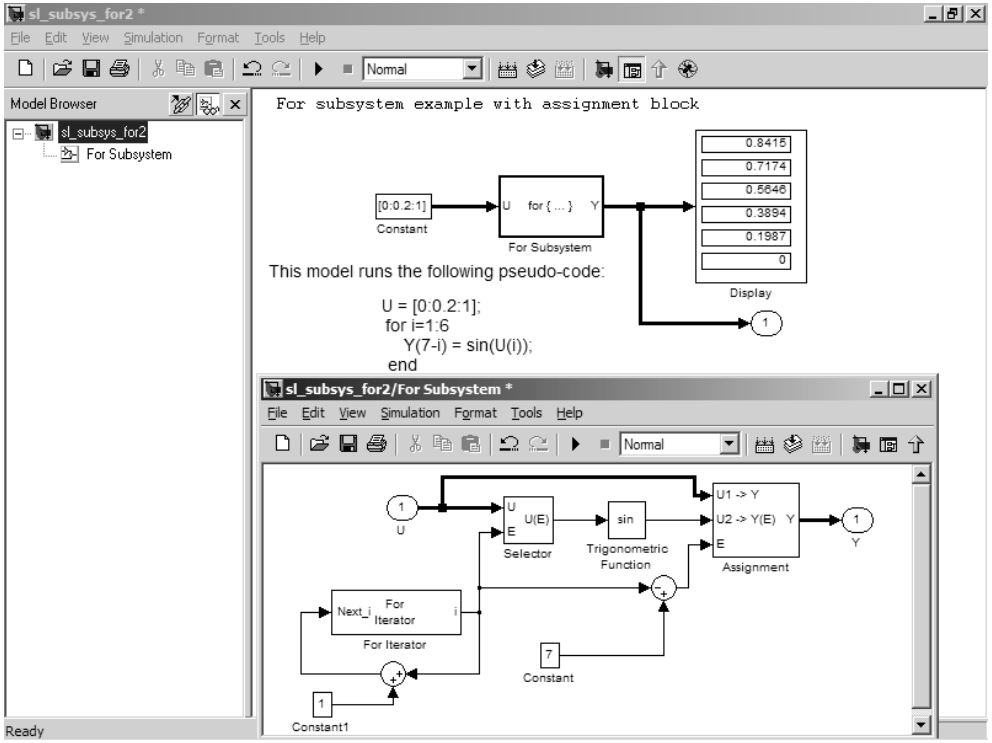


Рис. 8.27. Окно с примером применения подсистемы ***For-subsystem*** *после запуска примера*

- 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. урок 9);
- задать подсистеме статус маскированной подсистемы;

- с помощью специального редактора масок создать окно установки параметров, определить средства инициализации подсистемы, создать 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.

8.6.2. 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. 8.28. 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.

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.

8.6.3. 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. 8.29.

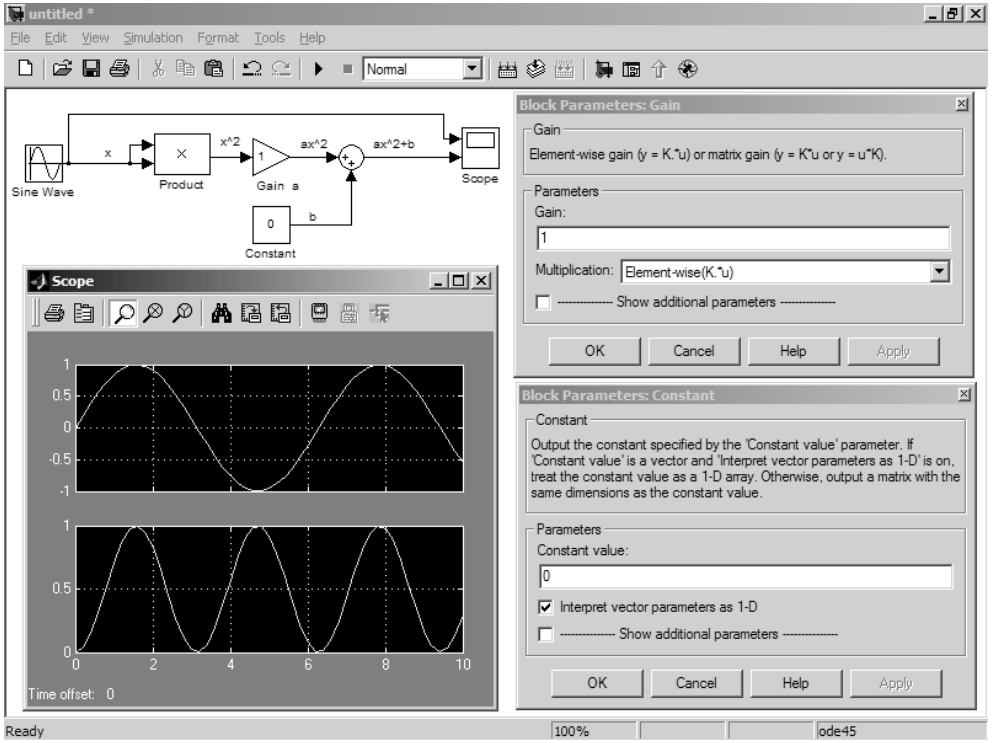


Рис. 8.28. Исходная модель

Выполнив команду меню **Edit ⇒ Create Subsystem**, создадим на базе выделенных блоков подсистему. Особенности этого процесса детально описывались выше, 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. 8.30).

После маскирования подсистемы графическая модель уже не будет выводиться. Двойной щелчок мышью на блоке будет выводить окно параметров блока. А для вывода графической модели придется использовать команду `Look under mask...` в позиции `Edit` меню окна Simulink.

8.6.4. 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** – рис. 8.31.

Данная команда запускает редактор маски – появляется его окно с открытой незаполненной вкладкой **Icon** (рис. 8.32). Редактор маски – это средство для создания полноценных библиотечных блоков, аналогичных профессионально раз-

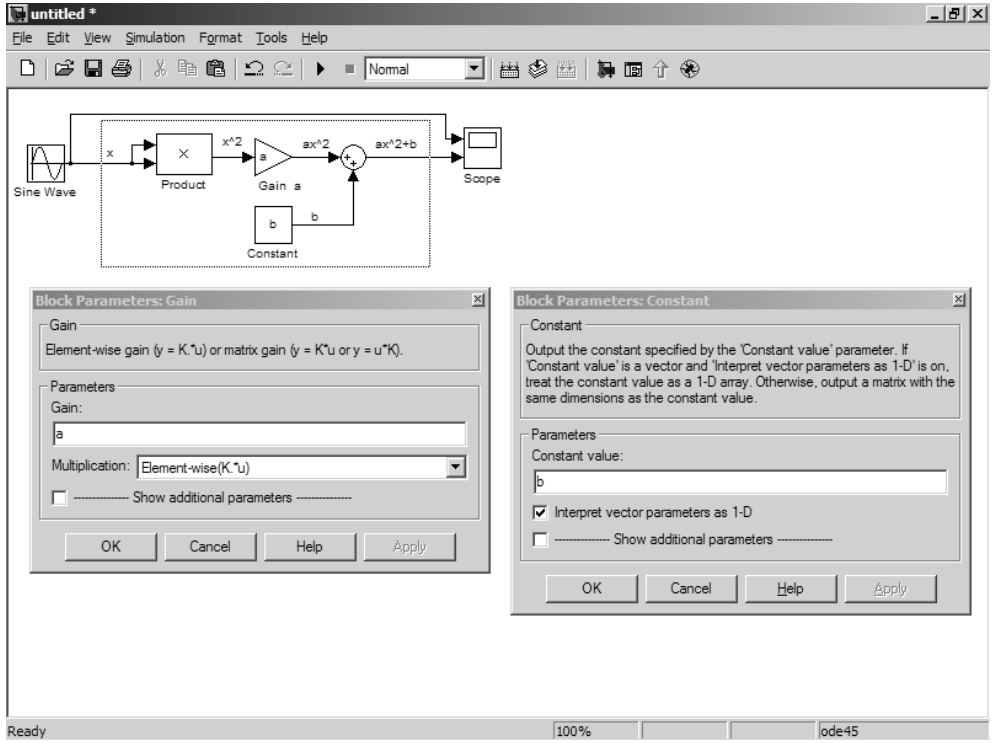


Рис. 8.29. Подготовка к созданию подсистемы

работанным блокам библиотеки Simulink. В версии Simulink 5/6 редактор маски существенно переработан в сравнении с редактором маски Simulink 4.

8.7. Работа с масками

8.7.1. Редактор маски

Как видно по рис. 8.32, редактор маски имеет четыре вкладки (в Simulink 4 их было 3). Отметим их назначение:

- **Icon** – подготовка пиктограммы (значка) блока;
- **Parameters** – подготовка окна параметров блока (появилась в Simulink 5);
- **Initialization** – подготовка средства инициализации;
- **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;

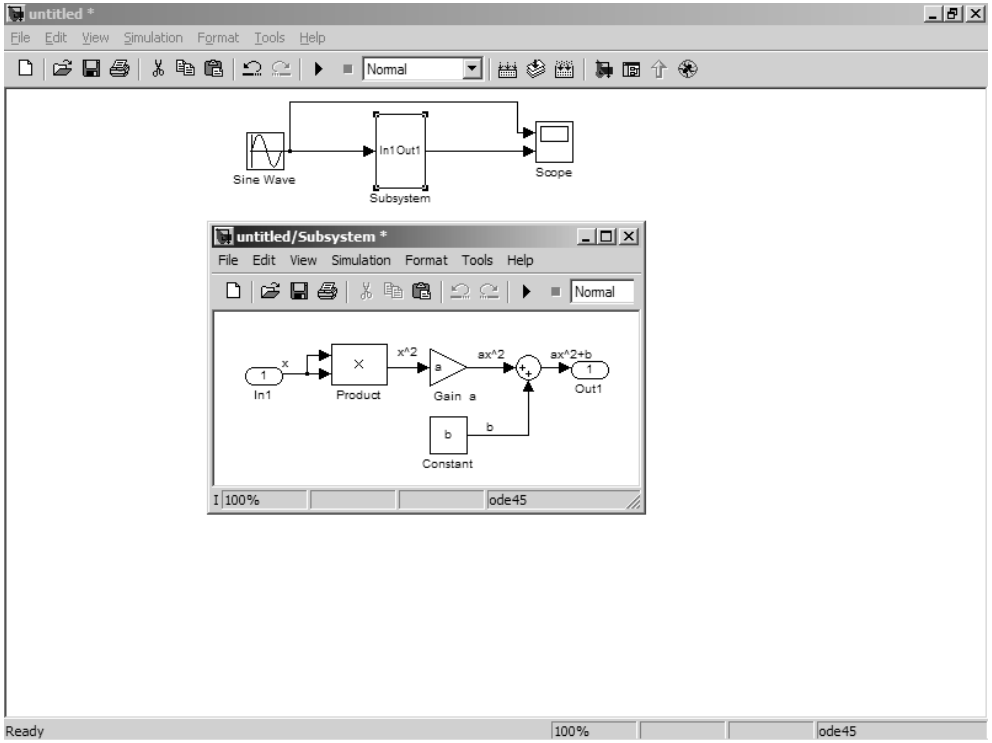


Рис. 8.30. Преобразованная модель с подсистемой

- **Unmask** – снять маску с выделенного блока;
- **Help** – открыть окно справки по редактору маски;
- **Apply** – применить заданные установки, не выходя из редактора маски.

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.

8.7.2. Создание окна параметров блока

Как отмечалось, окно редактора маски появляется с открытой вкладкой **Icon**. Однако большинство пользователей считают создание новой пиктограммы блока (или его значка) вовсе не первоочередным делом. Многих вполне устраивает исходный вид пиктограммы блока.

Куда важнее создать окно параметров блока. Оно может содержать параметры блока, а также флажки и списки. Для нашего блока **Quadrator** достаточно создать окно с двумя параметрами – *a* и *b*. Редактор маски позволяет задать до полутора

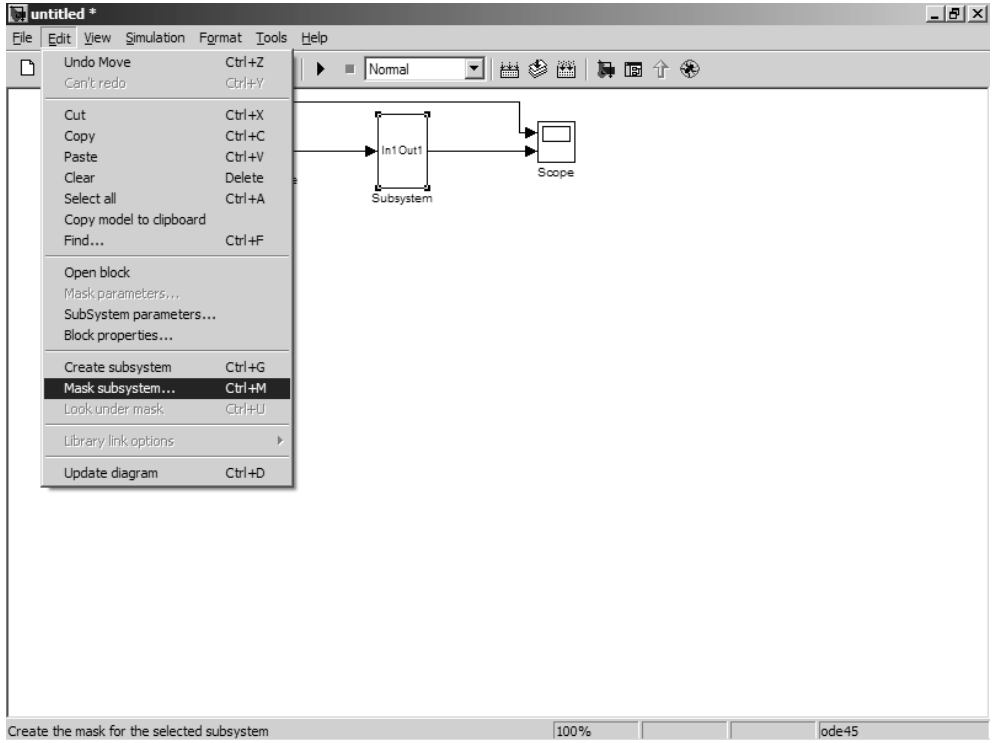


Рис. 8.31. Подготовка к запуску редактора маски

десятков параметров и элементов управления, причем разного вида (см. ниже). Для создания данных окна параметров служит вкладка **Parameters** (рис. 8.33).

При первом открытии вкладка **Parameters** содержит пустой список **Dialog parameters**. Этот список состоит из пяти столбцов:

- **Prompt** – имена используемых в маске блоков;
- **Type** – типы параметров (выбираются из раскрывающегося списка: **Edit** – редактируемое поле ввода, **Checkbox** – поле для флажка, **Popup** – раскрывающийся список);
- **Variable** – имена переменных, несущих значение соответствующих параметров;
- **Evaluate** – признак исполнения;
- **Tunable** – признак возможности изменения.

Для создания этого списка служат четыре кнопки вкладки, расположенные слева:

- **Add** – добавить в список новую позицию;
- **Delete** – стереть выделенную позицию списка;
- **Up** – поднять выделенную позицию списка вверх;
- **Down** – опустить выделенную позицию списка вниз.

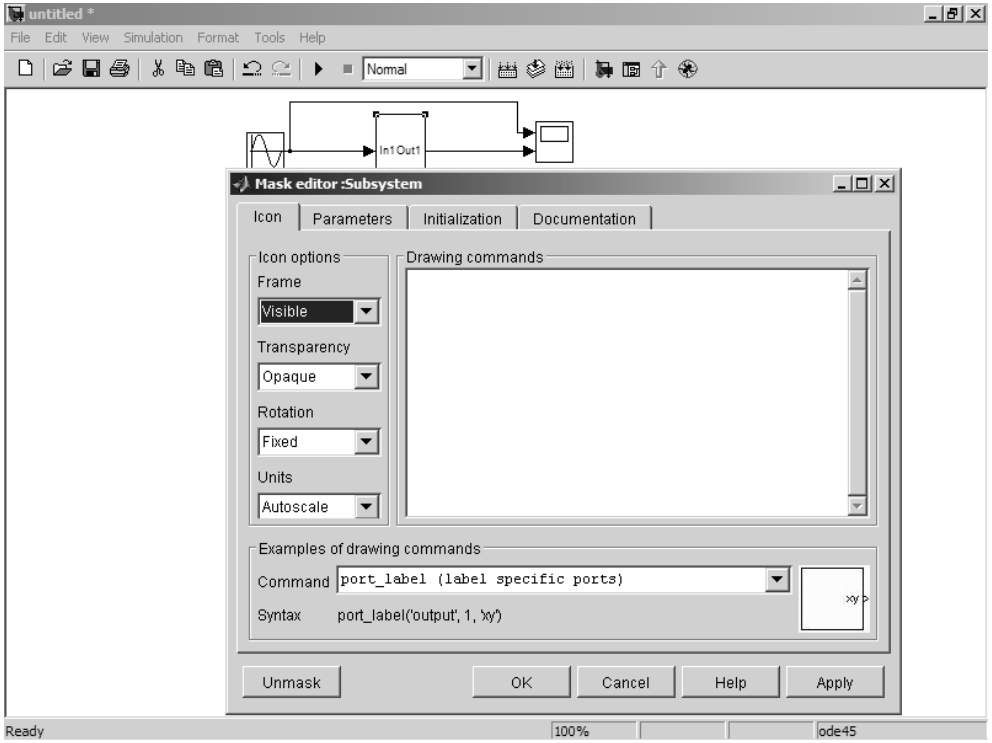


Рис. 8.32. Окно редактора маски после его запуска

В нашем случае, нажав кнопку **Add**, следует задать сначала имя блока **Gain** и переменную *a*, а затем имя блока **Constant** и переменную *b*. Для этих переменных надо использовать предложенные по умолчанию признаки исполнения и возможности изменения. Для завершения этого процесса достаточно нажать кнопку **Apply (Применить)**. Полезно обратить внимание на то, что возможно создание куда более сложных окон параметров со списками, окнами задания опций и иными деталями интерфейса.

8.7.3. Инициализация параметров

Вкладка **Initialization** используется, если это нужно, для задания средств инициализации блока. Эти средства действуют только внутри маскированной подсистемы. Если были определены переменные параметры, то их список появляется в поле Dialog Variables – см. рис. 8.34.

В нашем примере инициализация не требуется, и окно **Initialization commands** можно оставить пустым. Однако на самом деле возможности вкладки **Initialization** намного выше. В поле **Initialization Commands** можно записать любые выражения на языке системы MATLAB, в том числе содержащие условные выражения и переключатели. Это может потребоваться, например, для изменения вида пиктограммы создаваемого блока в зависимости от набора его параметров. После ввода нужных выражений надо не забыть зафиксировать это нажатием клавиши **Apply (Применить)**.

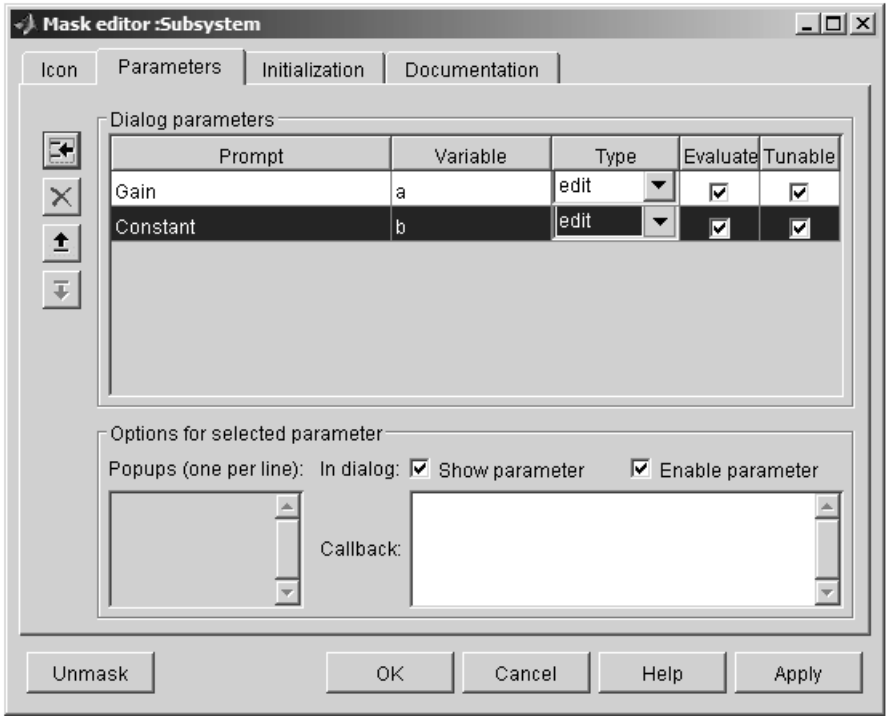


Рис. 8.33. Подготовка данных для окна задания параметров блока

Команды инициализации исполняются при открытии окна модели, при запуске модели на исполнение, выполнении команды **Update diagram** позиции **Edit** меню, при вращении или динамическом изменении пиктограммы.

8.7.4. 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. 8.35.

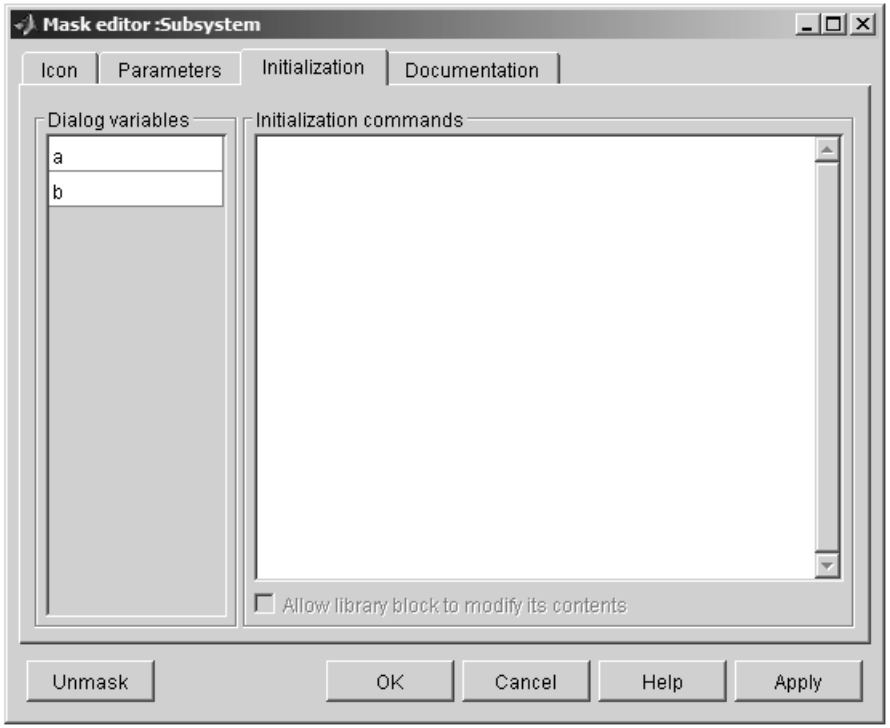


*Рис. 8.34. Работа с вкладкой **Initialization***

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.

8.7.5. 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 8.36 поясняет создание простейшей пиктограммы, которая пред-

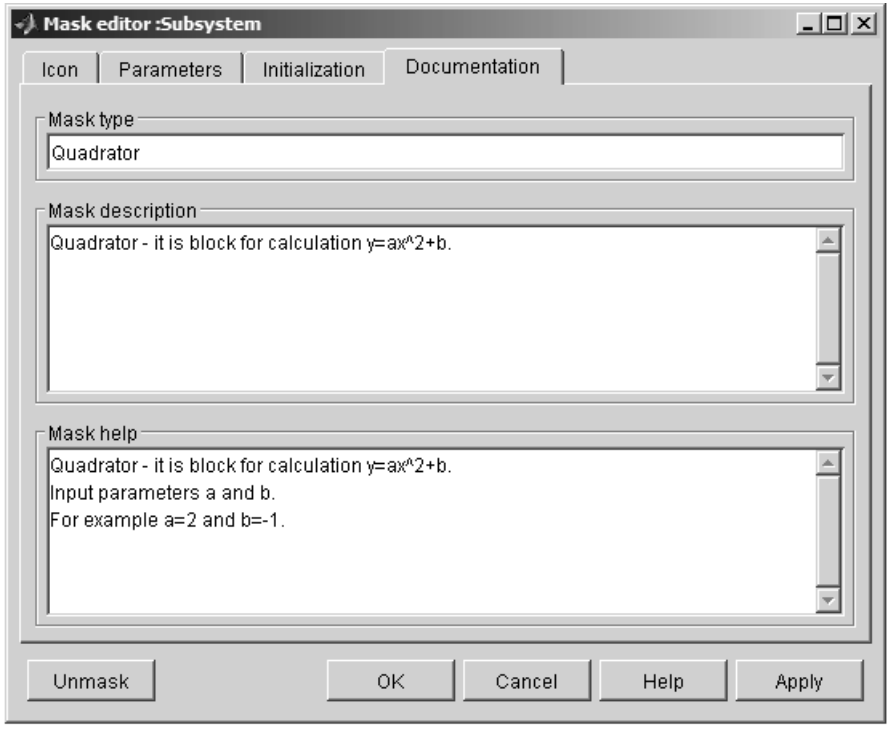


Рис. 8.35. Подготовка описания и справки по блоку

ставляет собой прямоугольник с надписью внутри, состоящей из двух строк: **Quadrator** и **y=ax^2+b**.

Как видно по рис. 8.36, 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`.

Пока для задания пиктограммы нашего блока этого достаточно. Нажав кнопку **Apply**, можно наблюдать превращение нашего блока в блок с новой пиктограммой (см. рис. 8.36 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.

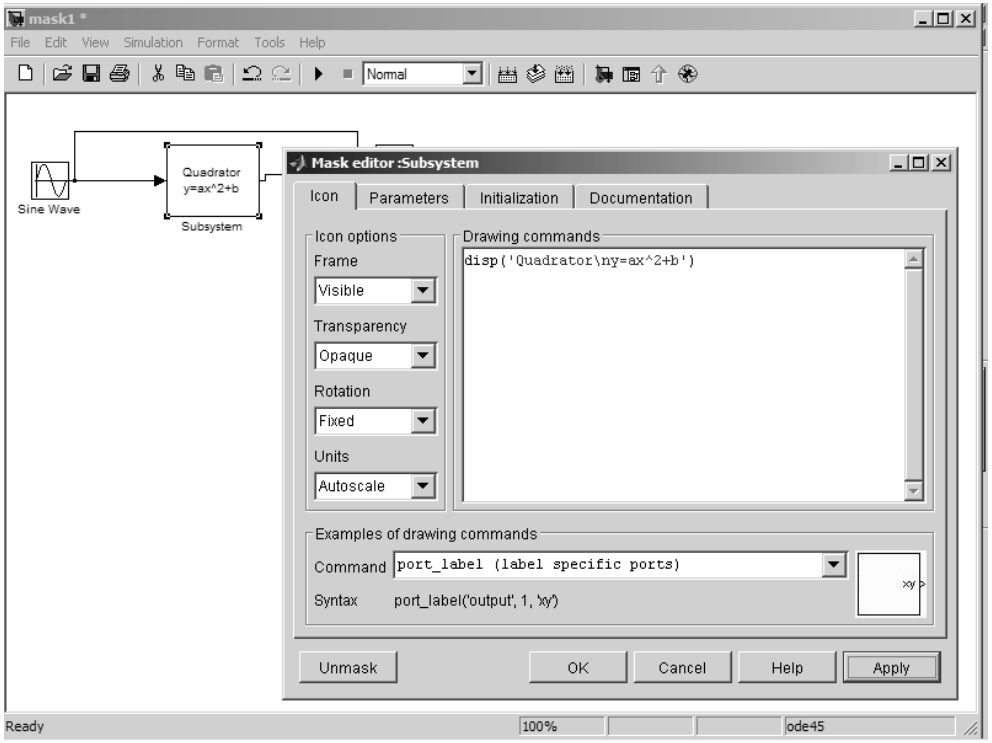


Рис. 8.36. Создание простой пиктограммы

8.7.6. Проверка модели с созданной маской

Если выполнить двойной щелчок мыши на созданной маскированной подсистеме (маске), то появится окно параметров маски (рис. 8.37).

Запустив модель кнопкой пуска моделирования, можно наблюдать работу модели и, в частности, появление осциллограмм. Сравнение их с осциллограммами исходной модели (рис. 8.28) показывает их полную идентичность, что свидетельствует о том, что созданная нами маскированная подсистема работает так, как это было задумано.

Теперь проверим, что модель с маскированной подсистемой может работать при других значениях параметров, например a=2 и b=-1. Введя в окно параметров подсистемы эти параметры и пустив модель на исполнение, получим приведенные на рис. 8.38 осциллограммы. Здесь выходной сигнал имеет те же пределы изменения, что и входной (от –1 до +1), и созданная подсистема работает почти как идеальный удвоитель частоты входного сигнала.

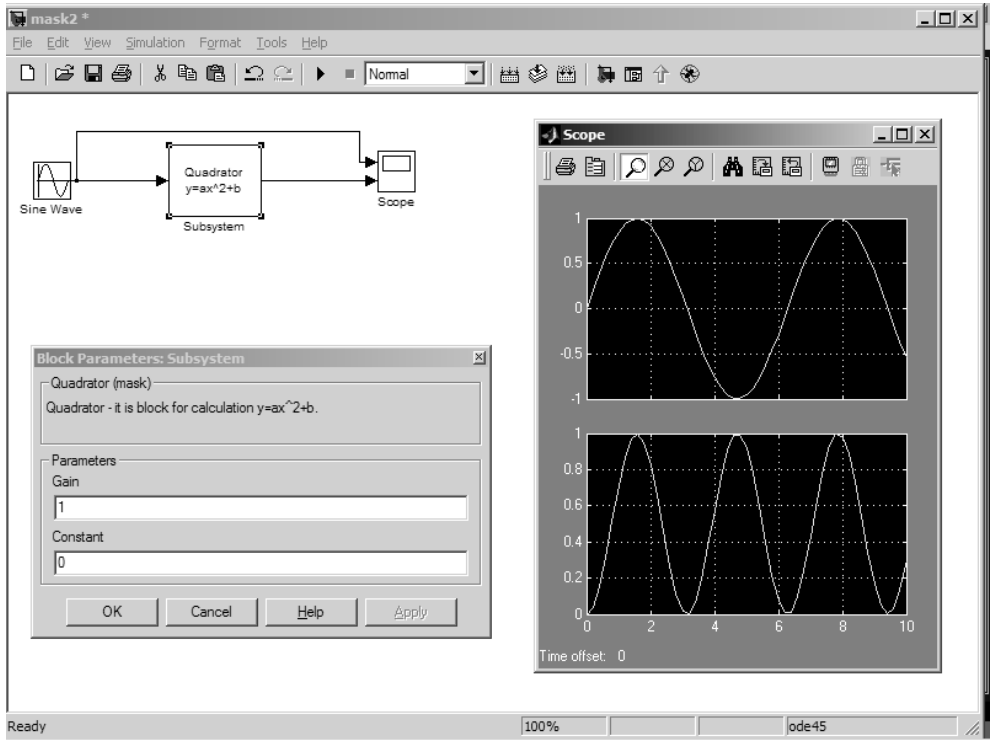


Рис. 8.37. Работа с моделью, имеющей маскированную подсистему (маску)

8.7.7. Вывод описания и справки маски

В верхней части окна параметров созданной маскированной подсистемы можно увидеть ранее созданное описание блока – см. рис. 8.37 и 8.39. А если нажать мышью кнопку **Help** в этом окне, то появится раздел стандартной справочной системы Simulink с текстом также ранее введенной справки (рис. 8.39).

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.

8.7.8. Создание маски-справки

На диаграммах моделей часто можно увидеть блоки со справочными данными. 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.

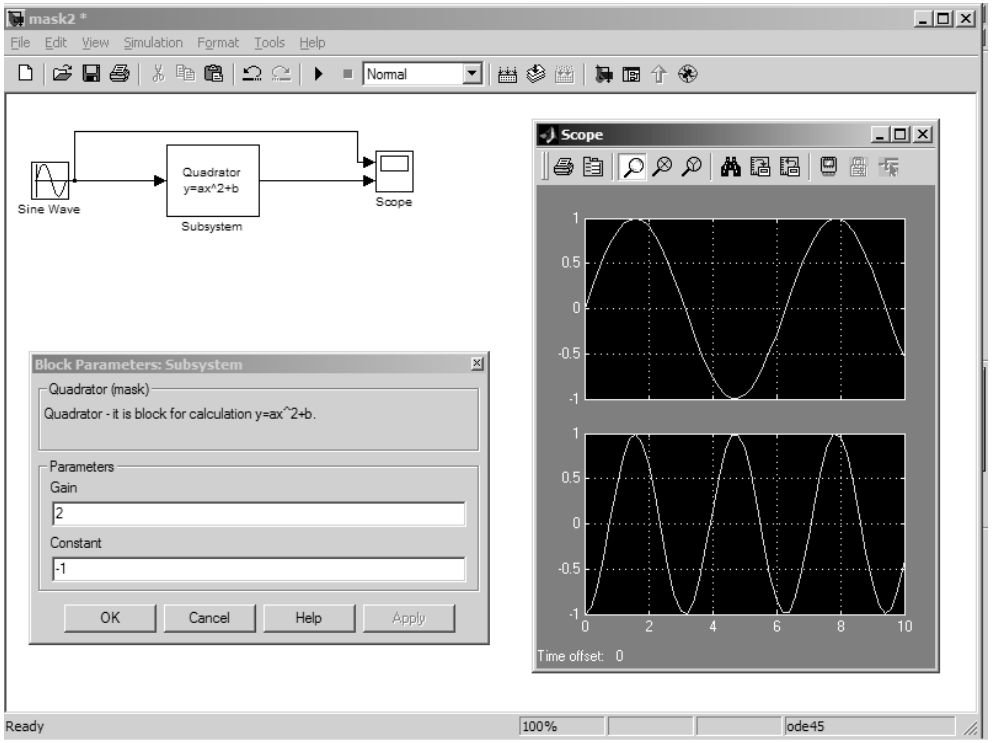


Рис. 8.38. Работа с моделью, имеющей маскированную подсистему (маску) после изменения параметров a и b

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.

8.8. Расширенные средства создания пиктограмм блоков

8.8.1. Задание текстовых надписей

Simulink вместе с базовой системой MATLAB открывает огромные возможности в создании любых пиктограмм. Но для этого необходимо 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

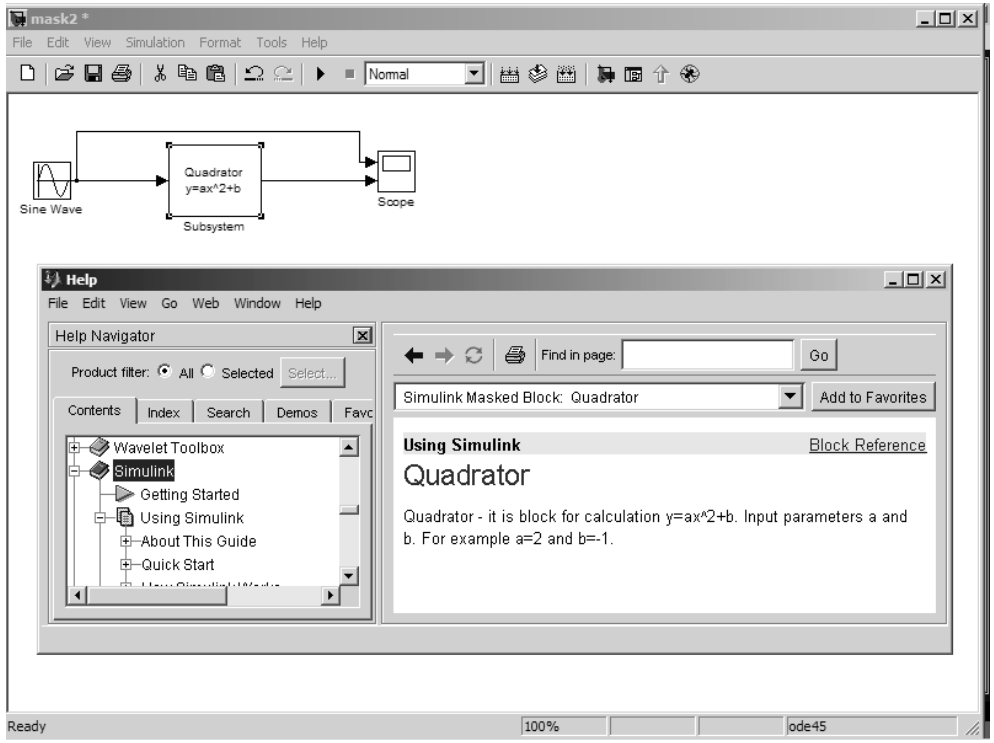


Рис. 8.39. Пример вывода описания и справки по маске

«золотую середину» в части возможностей оформления пиктограмм. Список этих команд открывает список **Command** на вкладке **Icon** окна редактора маски. Этот список определяет тот набор команд, которые можно использовать.

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**.

8.8.2. 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`. Формат команды можно найти, выбрав команду **plot** в списке **Command** редактора маски.

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:

```
plot([0.1,0.25,0.5,0.75,0.9],[1,0.3,0.1,0.3,1])
text(0.28,0.7,'y=ax^2+b')
```

Первая из них строит грубый график параболы по 5 точкам, а вторая задает вывод надписи. Нажав кнопку **Apply** окна редактора маски, можно тут же наблюдать изменение вида пиктограммы ранее созданной подсистемы – см. рис. 8.40.

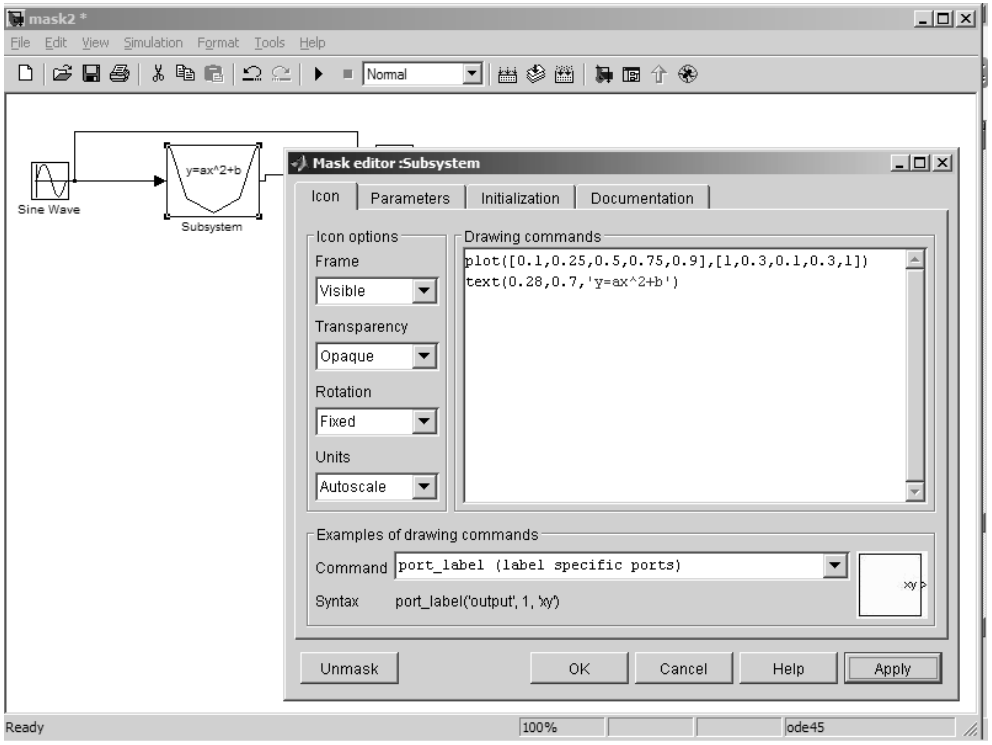


Рис. 8.40. Пример редактирования пиктограммы блока

Подбирая координаты точек в команде `plot` опытным путем, после нескольких попыток можно получить требуемое изображение параболы.

8.8.3. 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. 8.41 показан пример поворота на 90 градусов пиктограммы, созданной ранее. При таком повороте потребуется скорректировать связи между блоками.

Успех нашего опыта на рис. 8.41 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. 8.41 «уплыла».

Параметр **Normalized** задает нормализованное изображение. При этом контуры или рамка окна пиктограммы имеют не координаты левого нижнего угла (0,0), а координаты правого верхнего угла (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 пикселях. В этом случае решение задач (довольно непростых) по изменению масштаба и поворота рисунков пиктограмм возлагается целиком на пользователя.

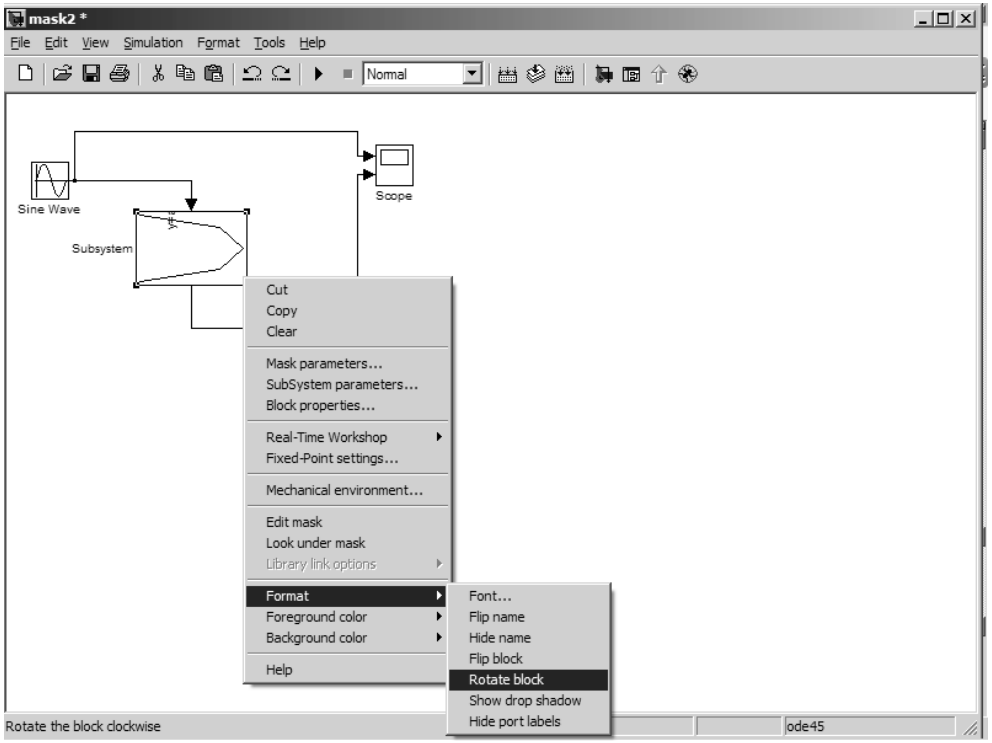


Рис. 8.41. Пример поворота пиктограммы

8.8.4. 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. 8.42).

При этом запрашивается имя диаграммы модели и блока, для которого создается пиктограмма. Данный диалог виден на рис. 8.42. 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. 8.42 снизу).

Работа с редактором элементарна. Сначала имеется пустое окно редактирования рисунка с крестом во весь экран из черных линий. Для создания рисунка по точкам используем 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.

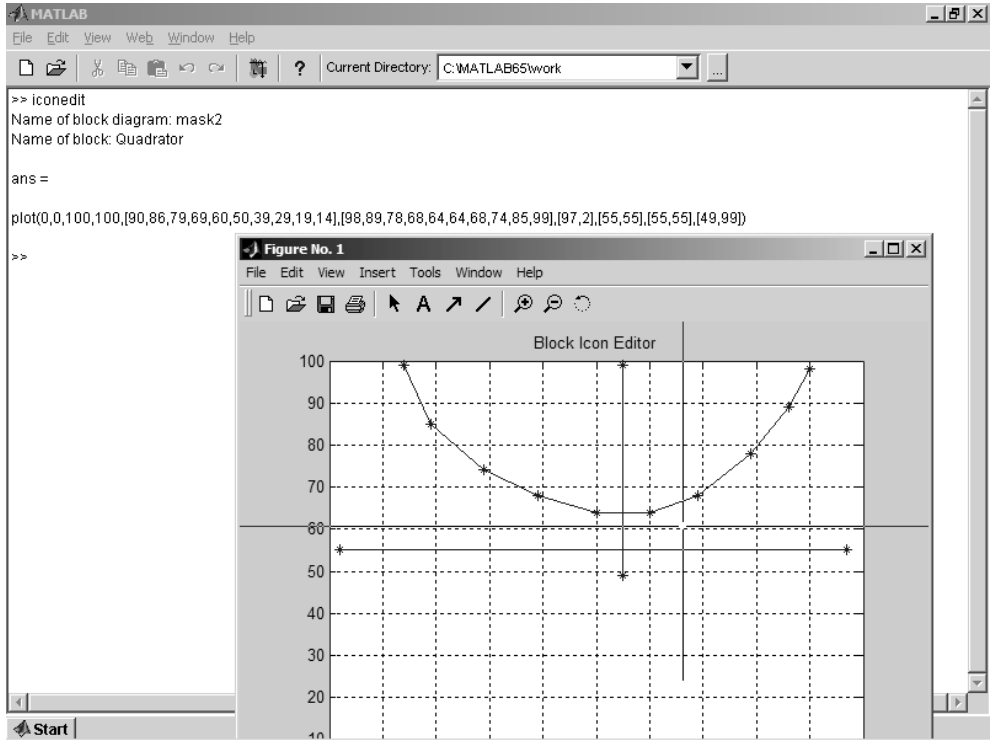


Рис. 8.42. Пример работы с графическим редактором пиктограмм

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. 8.42 показан пример довольно грубого построения параболы с координатными осями.

По завершении работы с редактором пиктограмм его окно закрывается нажатием клавиши `q`, и в командной строке MATLAB появляется графическая команда, обеспечивающая построение заданного графика. Она появляется также в окне редактора маски в поле **Drawing commands** вкладки **Icon**. Можно также добавить команду вывода текстового комментария. Все это показано на рис. 8.43.

Рисунок 8.44 иллюстрирует поворот созданной пиктограммы с коррекцией связей между блоками и установленным параметром **Rotate**. Как нетрудно заметить, на этот раз можно получить развернутую пиктограмму с правильным расположением надписей внутри нее. Хотя внешняя надпись осталась расположенной горизонтально.

Разумеется, можно отредактировать команды графики на вкладке **Icon**. Например, на рис. 8.45 представлен еще один вариант построения пиктограммы для

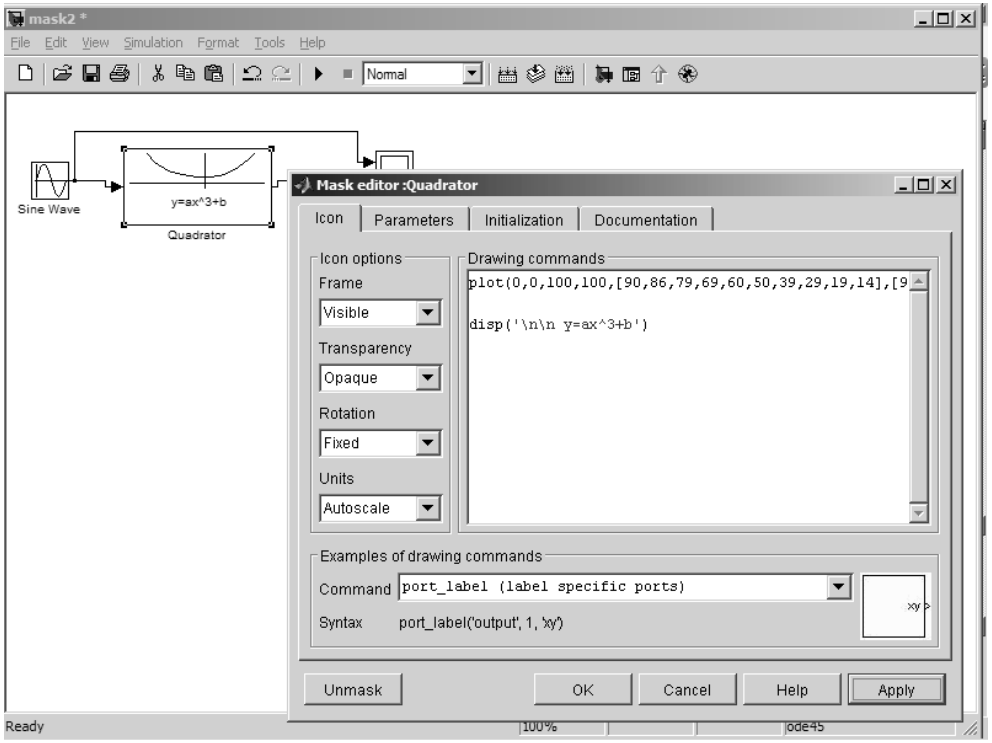


Рис. 8.43. Коррекция пиктограммы

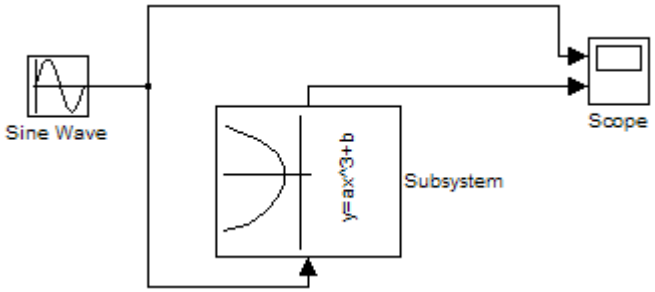


Рис. 8.44. Пример поворота пиктограммы, созданной с помощью редактора пиктограмм

нашего примера. Обратите внимание на открытый список команд, которые можно использовать для построения пиктограмм в окне редактора маски.

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.

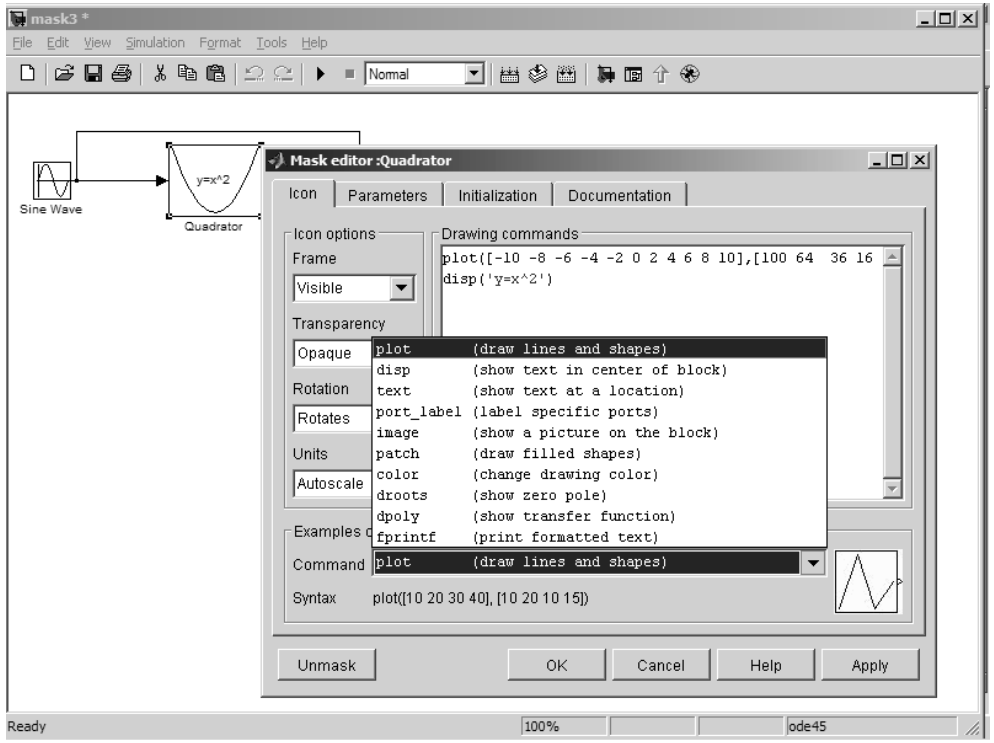


Рис. 8.45. Пример применения команд из списка команд редактора маски

8.8.5. 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 подобного файла служит графическая команда:

```
image(imread("имя_файла", "тип_файла")
```

Пример создания пиктограммы на основе применения данной команды показан на рис. 8.46.

В данном случае в качестве рисунка взято изображение самолета.

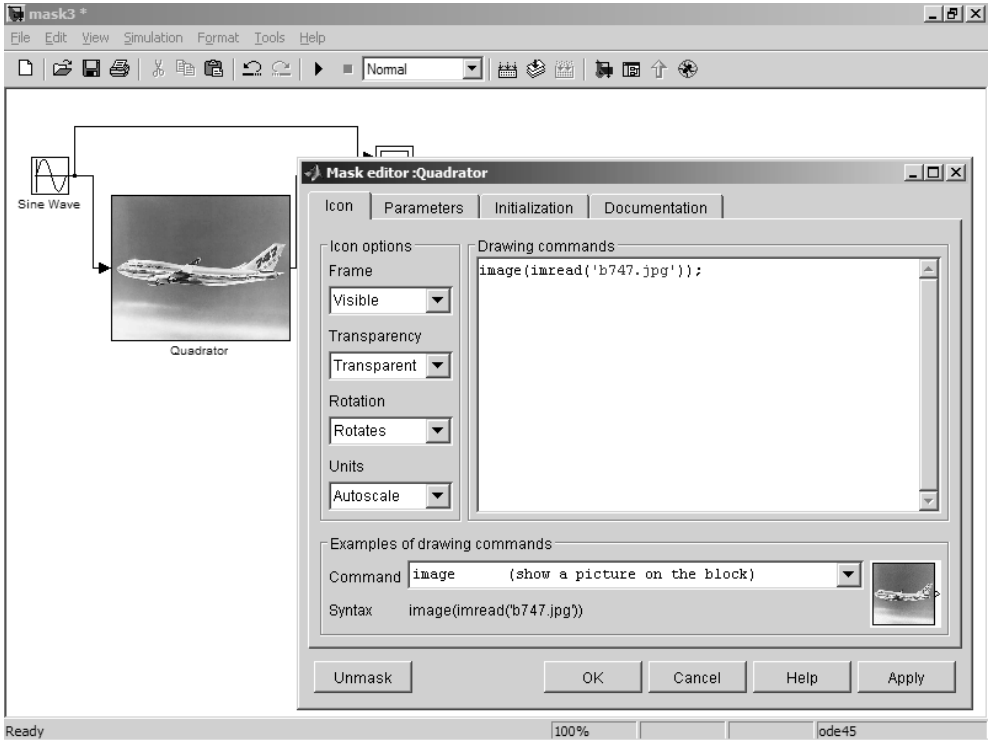


Рис. 8.46. Создание пиктограммы на основе графического файла

8.9. Создание библиотек пользователя

8.9.1. Библиотека Commonly Used Simulink 6

При серьезной работе с Simulink может потребоваться создание библиотек пользователя, составленных как из созданных им блоков, так и из 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. В Simulink 6, учитывая это, была даже создана специальная библиотека наиболее распространенных блоков **Commonly Used**.

В состав этой библиотеки вошли следующие блоки:

```
Bus Creator             Bus Selector        Constant
Data Type Conversion    Demux               Discrete-Time Integrator
Gain                    Ground              Inport
Integrator              Logical Operator    Mux
Outport                 Product             Relational Operator
```



```
Saturation              Floating Scope      Signal Viewer Scope
Subsystem               Atomic Subsystem    CodeReuse Subsystem
Sum                     Switch              Terminator
Unit Delay
```

Все эти блоки входят в другие библиотеки Simulink 5/6 и уже были описаны. Так что их включение в состав библиотеки Commonly Used связано не с новизной данных блоков, а лишь с желанием разработчиков дать пользователю небольшую библиотеку «на все случаи жизни». Увы, даже этот «подарок» не всегда позволяет обходиться только встроенными блоками.

8.9.2. Требования к библиотекам пользователя

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)**. Уже в Simulink 4.0 эта команда была исключена.

Библиотека, или набор блоков, пользователя может быть создана в специальном окне библиотек пользователя. Это окно открывается командой **File ⇒ New ⇒ Library** (рис. 8.47).

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.

8.9.3. 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.

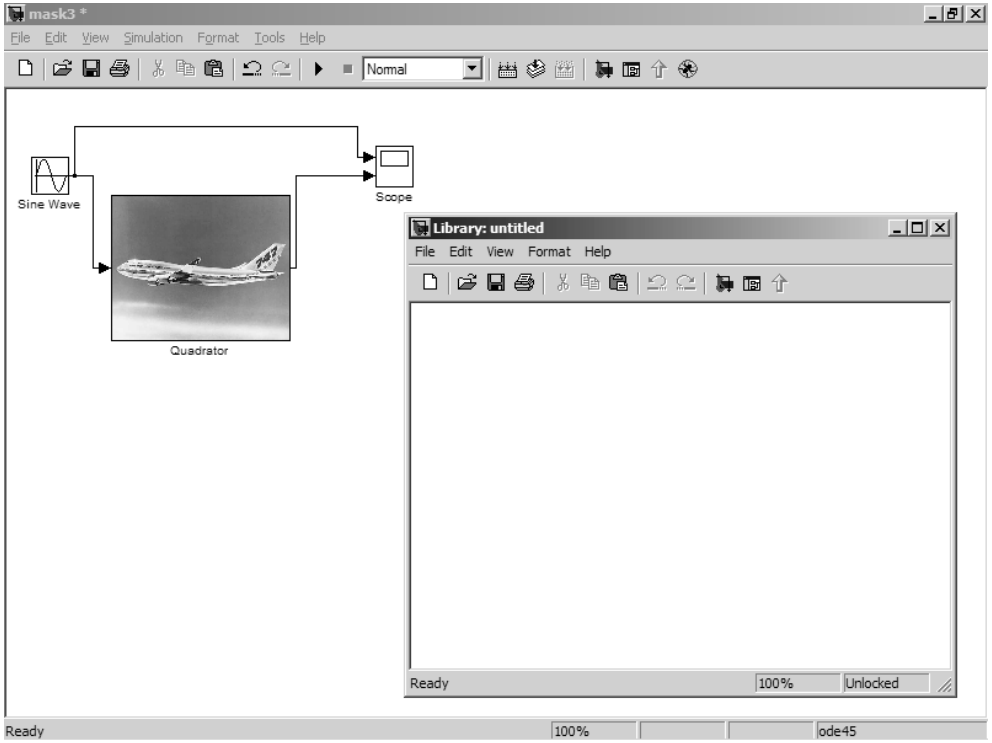


Рис. 8.47. Создание окна библиотеки пользователя

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 8.48 показывает созданную таким образом новую библиотеку **mylib**.

После заполнения блоками новая библиотека блокируется командой меню **Edit ⇒ Locked**. После этого библиотека становится недоступной для пополнения и модернизации. Впрочем, можно разблокировать 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.

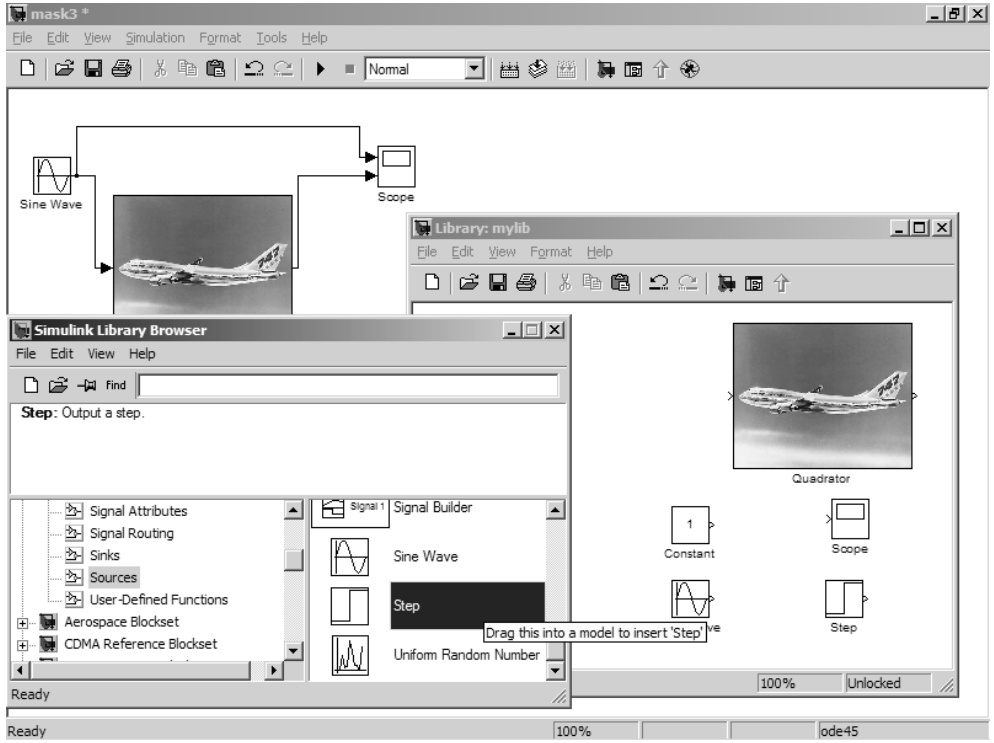


*Рис. 8.48. Ссозданная новая библиотека **mylib***

8.9.4. 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.

Урок 9

Инструменты и практика моделирования

Даже простые модели систем и устройств могут не работать или работать неверно. Причинами этого могут быть как неверное построение модели, так и неверный метод и параметры моделирования или неудачный выбор параметров отдельных блоков модели. Для отладки и испытания моделей в Simulink входят специальные инструменты, которые изучаются в этом уроке.

9.1. Меню инструментов Tools

9.1.1. Роль инструментальных средств Simulink

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 модель (диаграмму) 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 или иные графические зависимости. Ставшие ненужными регистраторы можно убрать из окончательной модели.

Тем не менее Simulink имеет и 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.

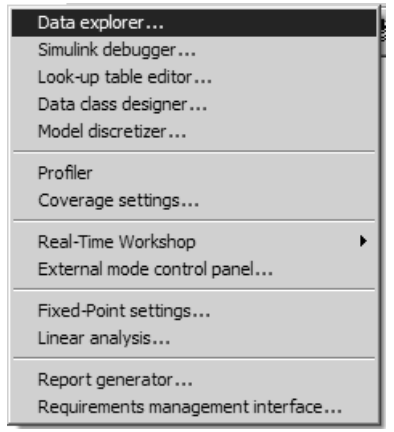
9.1.2. Меню инструментов Tools

Основные инструментальные средства Simulink 5/6 сосредоточены в новом меню – **Tools**. Его состав для Simulink 5 показан на рис. 9.1. Появление этого меню облегчает доступ к разнообразным инструментальным средствам Simulink, которые теперь собраны воедино.

Меню **Tools** имеет ряд команд:

- `Data Explorer...` – открытие браузера данных;
- `Simulink Debugger...` – открытие отладчика моделей Simulink;
- `Look-up table editor...` – открытие окна редактора Look-up таблиц;
- `Data class editor...` – открытие окна редактора класса данных;
- `Model discretizer...` – открытие окна дискретизатора моделей;
- `Profiler` – открытие окна профилирования моделей;
- `Coverage Setting...` – открытие окна установок представления отчета по моделированию;
- `Real Time Workshop` – доступ к расширению Real Time Workshop;
- `External mode control panel...` – вывод панели контроля моделирования при внешнем управлении;

*Рис. 9.1. Меню **Tools** окна моделей Simulink 5*



- `Fixed Point setting...` – доступ к расширению Fixed point;
- `Linear analysis...` – открытие окна линейного анализа;
- `Report generator...` – открытие генератора отчетов;
- `Requirements management interface...` – открытие окна просмотра модели.

Число позиций меню в Simulink 6 (особенно 6.6) заметно возросло. Это видно по рис. 2.27, где эта позиция меню для Simulink 6.6 показана с раскрытым списком команд. Далее мы рассмотрим инструментальные средства Simulink 5/6 в порядке важности их применения большинством пользователей.

9.2. Работа с отладчиком графических S-моделей

9.2.1. 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...**;
- 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. 9.2). Оно приведено для расширения Simulink 5.

9.2.2. Панель инструментов отладчика

Окно отладчика не имеет меню – управление им происходит только с помощью кнопок панели инструментов (рис. 9.3).

Эта панель имеет следующие органы управления (в основном кнопки):

1. **Step into current method** – шаг в текущий метод;
2. **Step over current method** – шаг с пропуском текущего метода;
3. **Step out of current method** – шаг из текущего метода;
4. **Step to first at start of next start time** – шаг к началу следующего стартового времени;

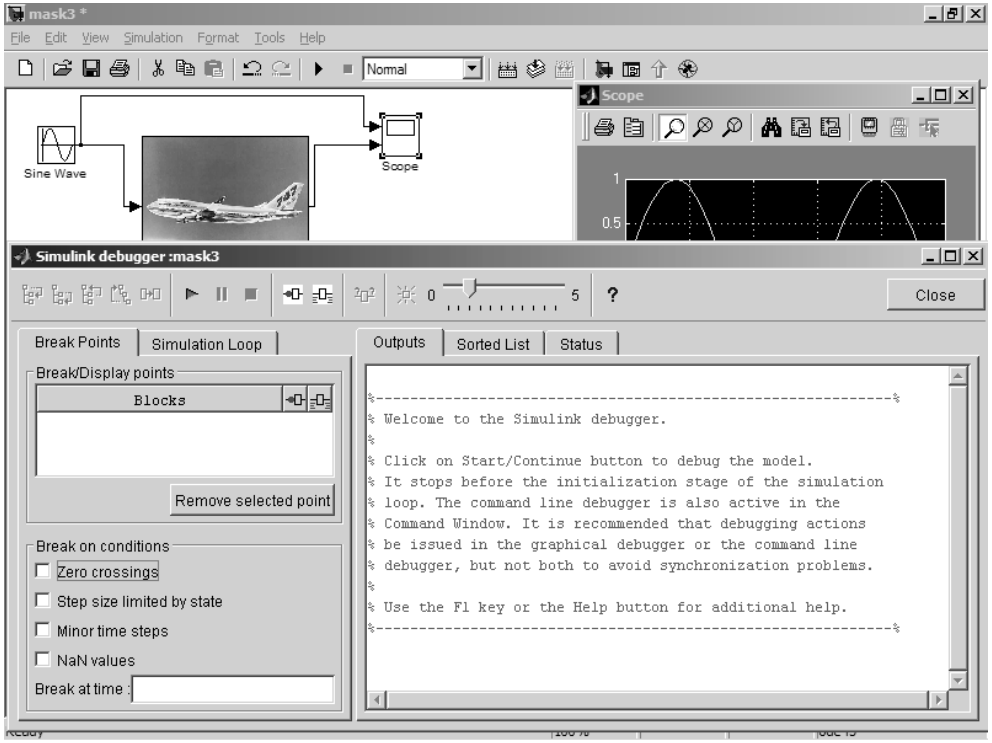


Рис. 9.2. Начало работы с отладчиком моделей

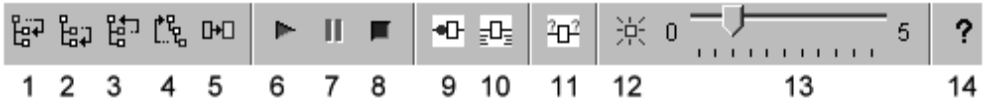


Рис. 9.3. Панель инструментов отладчика моделей

5. **Go to start of next block method** – переход к началу следующего метода блока;
6. **Start/Continue** – старт/продолжение отладки;
7. **Pause** – пауза в отладке;
8. **Stop debugging** – остановка отладки;
9. **Break before selected block** – установка точки прерывания перед выделенным блоком;
10. **Display I/O of selected block with executed** – показ исполняемых блоков ввода/вывода;
11. **Display current I/O of selected block** – показ текущих блоков ввода/вывода;
12. **Enable/Disable Animation** – подключение/отключение анимации;
13. **Animation Delay** – регулятор времени задержки анимации;
14. **Help** – вызов справки по отладчику.

9.2.3. 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 размещены короткое приглашение к работе с отладчиком и пояснение его назначения (рис. 9.2).

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**. Пример включения блока `Sine Wave` в список точек останова показан на рис. 9.4. Для каждого блока можно задать опции останова и показа его выходного сигнала (установкой флажка после имени блока в списке).

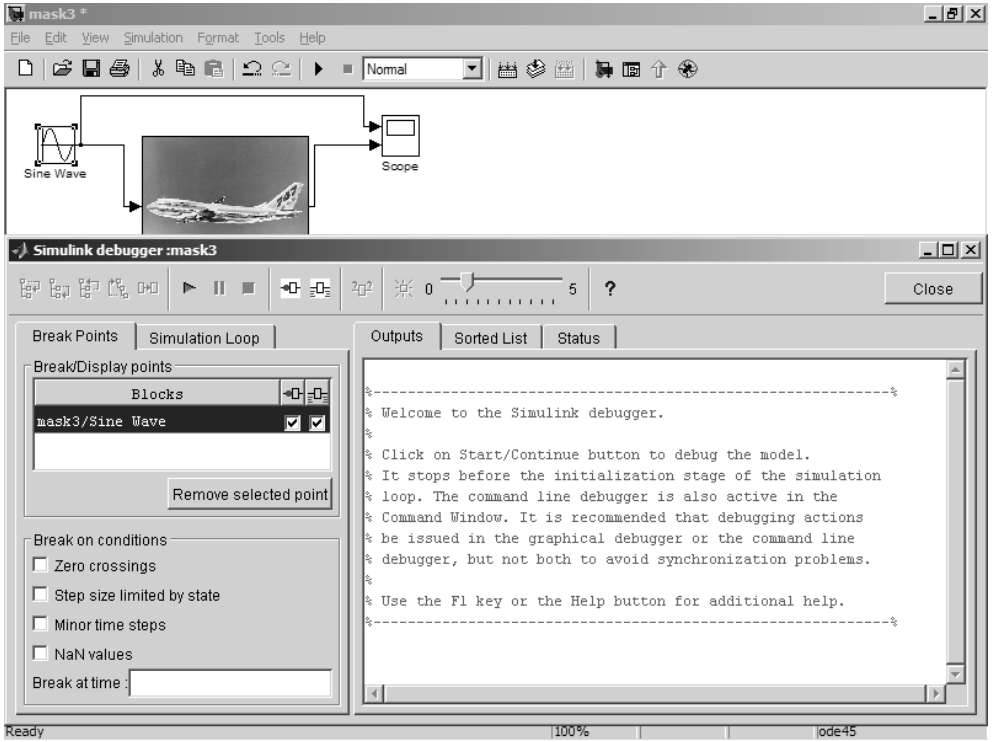


*Рис. 9.4. Включение блока **Sine Wave** в список точек останова*

При запуске модели будет происходить остановка моделирования перед каждым из блоков, включенных в список **Blocks**, с выводом результатов на каждом шаге. Признаком работы отладчика является появление в верхнем левом углу

окна модели информационного сообщения, выделенного прямоугольником из пунктирных черных линий. При этом блок, на котором произошла остановка моделирования, указывается линией, выходящей из блока информационного сообщения и кончиком указывающей на блок линии (ранее блоки выделялись цветом). Если линия указывает на обычный блок, то она принимает вид стрелки, цвет которой меняется в зависимости от работы блока (например, активизации входного или выходного порта). Для удаления выделенных блоков из списка **Blocks** служит кнопка **Remove select points**.

На рис. 9.5 дан пример пошаговой работы модели, в котором контролируются несколько блоков модели. Виден момент проверки выхода блока `Sine Wave`. Для выполнения очередного шага отладки следует нажать кнопку **Start/Continue** панели инструментов.

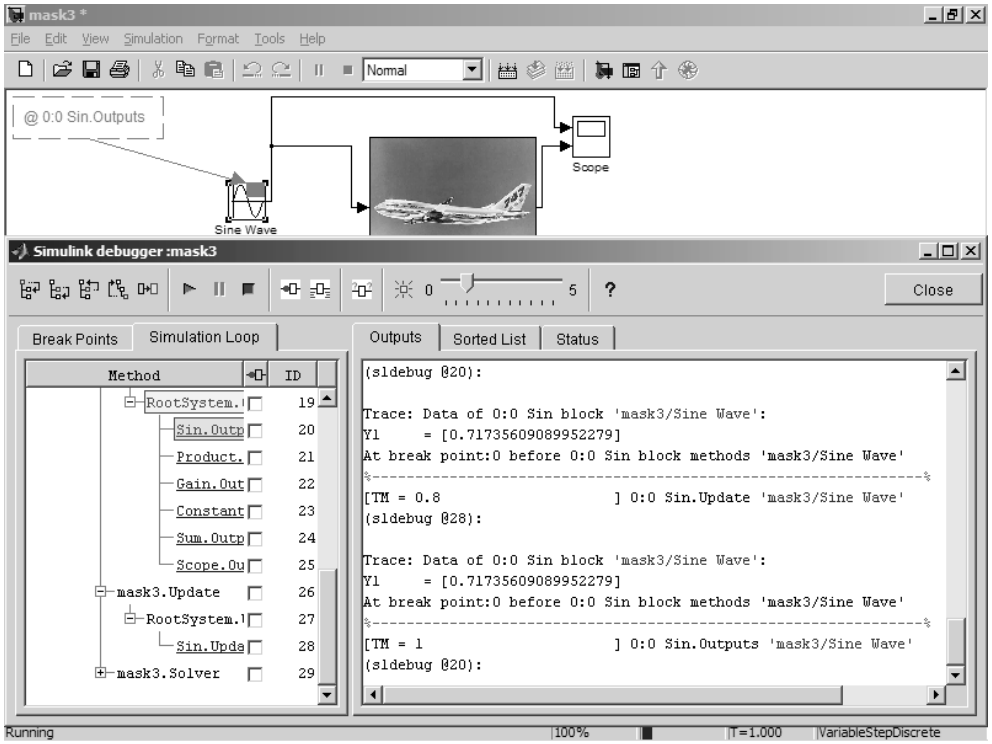


Рис. 9.5. Момент тестирования блока ***Sine Wave***

Как видно по рис. 9.5, на вкладке **Outputs** отражаются 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 $\mathbf{U}_i$ и выходного $\mathbf{Y}_i$. Их анализ позволяет оценить корректность работы блока.

В ходе отладки блоки из списка тестируются последовательно. Если в данный момент тестируется блок, расположенный в подсистеме (в том числе маскированной), то в информационном окне появляется информация о том, какой блок подсистемы тестируется в данный момент.

Помимо кнопки последовательной пошаговой отладки по списку **Blocks**, можно воспользоваться и другими кнопками в панели инструментов. Например, 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.

9.2.4. 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.

Можно просмотреть весь цикл отладки в режиме анимации. Для этого достаточно активизировать кнопку `Animation` и выставить регулятором (или оставить по умолчанию) скорость анимации.

9.2.5. Проверка порядка выполнения блоков

Вкладка `Sorted List` (в отладчике Simulink 4 она называлась `Execution Order`) окна отладчика позволяет получить информацию о порядке выполнения блоков в ходе моделирования (рис. 9.6).

Содержание этой вкладки очевидно. Обратите внимание лишь на то, что в маскированных подсистемах отражаются все входящие в них блоки. На рис. 9.6 представлен момент, когда тестируется выход блока **Product**, входящего в маскированную подсистему Quadrator (об этом можно судить по надписи в информационном блоке отладчика и линии, указывающей на подсистему Quadrator).

9.2.6. Оценка состояния отладчика

Состояние отладчика на текущем шаге моделирования можно оценить по информации, приводимой на вкладке **Status** (рис. 9.7). 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 трас-

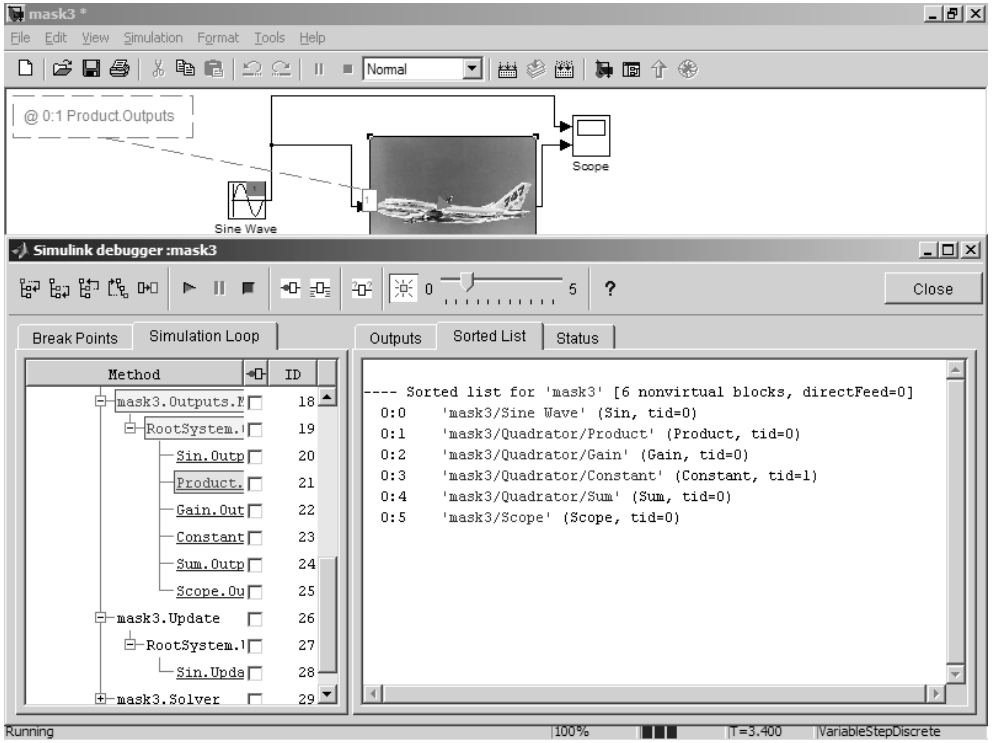


Рис. 9.6. Просмотр порядка выполнения блоков

сировки и др. Всего используется около полутора десятков признаков состояния, и все они видны на упомянутой вкладке.

9.2.7. 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

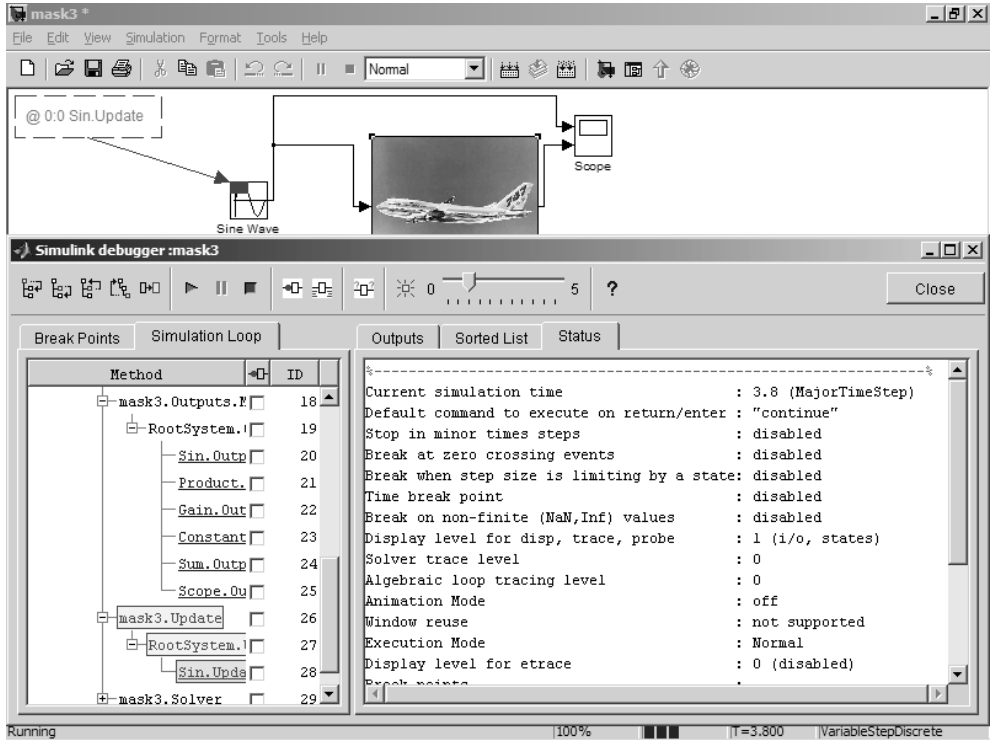


Рис. 9.7. Оценка состояния отладчика

Команда	Краткая форма	Повтор	Описание
`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 показа 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
`quit`	`q`	No	Прерывание моделирования
`Run`	`R`	No	Запуск моделирования

Команда	Краткая форма	Повтор	Описание
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. 9.8. В первой строке видана команда запуска отладчика из командной строки: `sldegug('mask3')`. 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. Для завершения работы с отладчиком можно использовать комбинацию клавиш `Ctrl` и `C`.

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.

9.3. Браузер данных Simulink

Для контроля данных, которые используются пакетом Simulink, служит браузер данных. Его окно (рис. 9.9) вызывается при помощи команды меню **Tools ⇒ Data Explorer...**. В данном случае данные относятся к модели демонстрационного примера f14.

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.

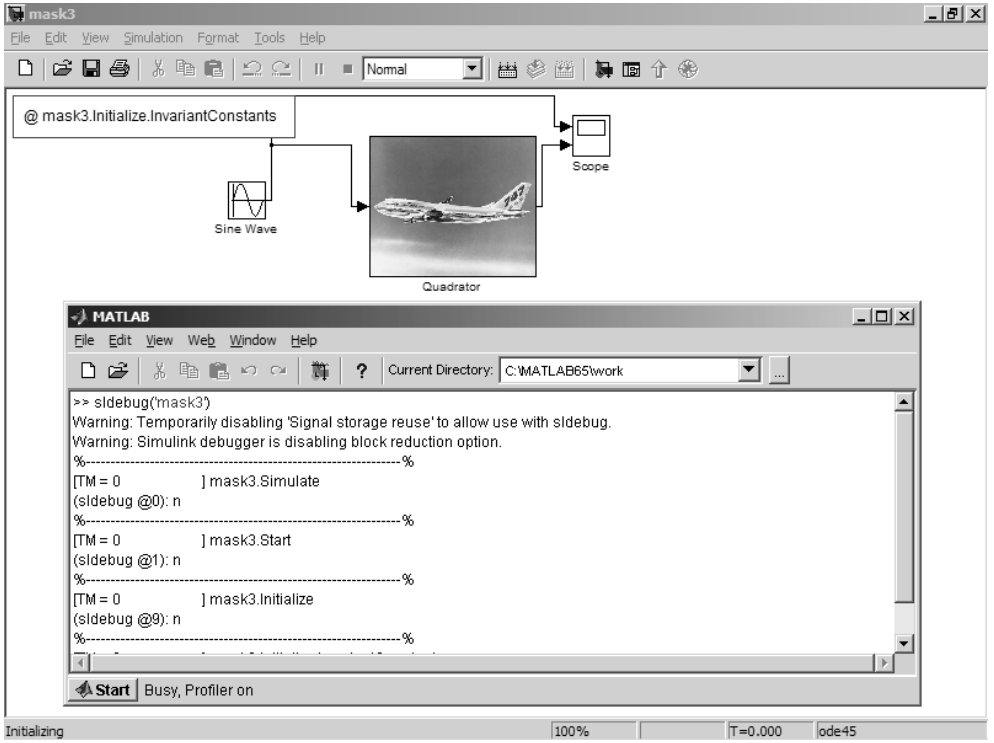


Рис. 9.8. Работа с отладчиком S-моделей из командной строки MATLAB

9.4. Подготовка отчетов по моделированию

9.4.1. 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 отчетов*. В Simulink 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.

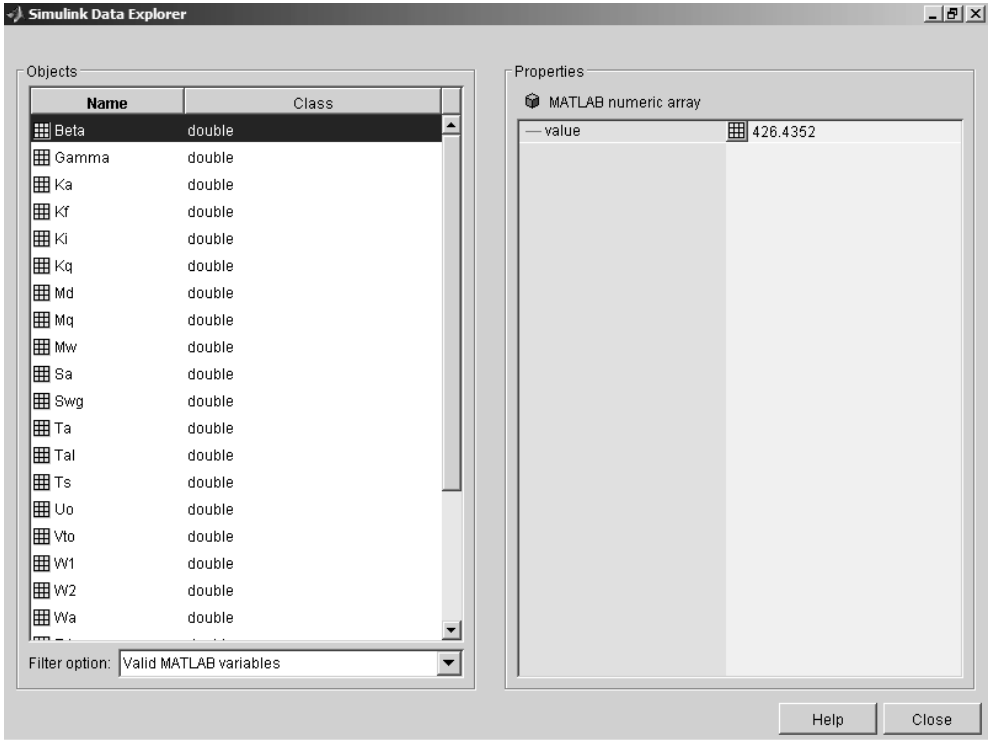


Рис. 9.9. Окно браузера данных Simulink

9.4.2. Установки просмотра отчета

Simulink обеспечивает детальный анализ данных моделирования с помощью специального отчета. Команда меню **Tools ⇒ Coverage Setting...** выводит окно с установками параметров просмотра отчета (рис. 9.10).

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. 9.10 слева). Можно также задать запись объектов в рабочую область, автоматическое увеличение индекса в имени переменных и задание генерации отчетов в виде HTML-файлов.

На рис. 9.11 показана вкладка **Report** окна **Coverage Setting** (справа). Здесь особо надо отметить опцию **Generate HTML Report (Сгенерировать HTML-отчет)**, которая задает возможность генерации отчета на языке программирования HTML, принятого в Интернете в подготовке гипертекстовых документов и справок. Кнопка **Setting...** открывает окно установок HTML-отчета, также показанное на рис. 9.11 слева. Ввиду того, что лимит на объем данной книги практически

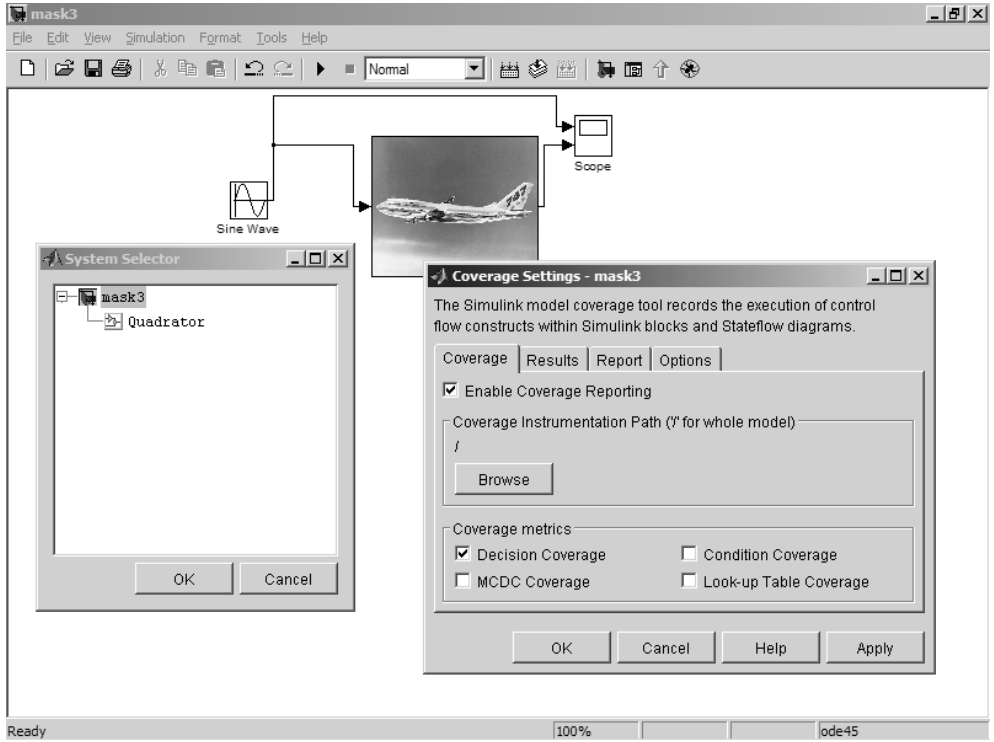


Рис. 9.10. Окно настройки отчетов

исчерпан, изучение несложных опций этого окна и других вкладок окна **Coverage Setting...** оставим читателю для самостоятельной разборки.

9.4.3. Запуск генератора отчетов

Для запуска генератора отчетов надо выполнить команду меню **Tools ⇒ Report generator...**. Появится окно генератора отчетов, представленное на рис. 9.12.

В этом окне имеется множество файлов-шаблонов для подготовки отчетов. Чтобы формировать отчет по шаблону, выберите требуемый шаблон и нажмите кнопку **Set**. Модель будет запущена на выполнение автоматически, и все стадии ее выполнения будут отражены в автоматически создаваемом отчете.

9.4.4. 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. 9.13.

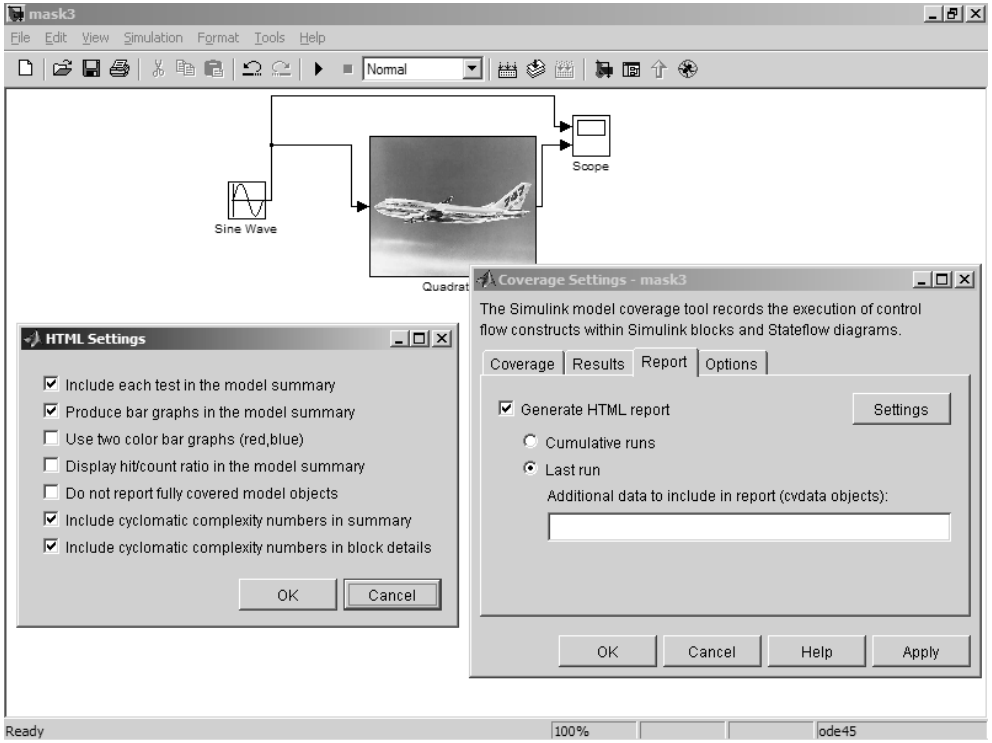


*Рис. 9.11. Окно **Coverage Setting** с открытой вкладкой **Record***

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.

9.4.5. Пример подготовки отчета

Для подготовки отчета по текущей модели (например, созданной ранее простой модели mask3) достаточно запустить интернет-браузер и 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** (рис. 9.10) установить флажок **Generate HTML report**.

На рис. 9.14 показано начало отчета в окне браузера Internet Explorer 6.0, входящем в состав операционной системы Windows XP.

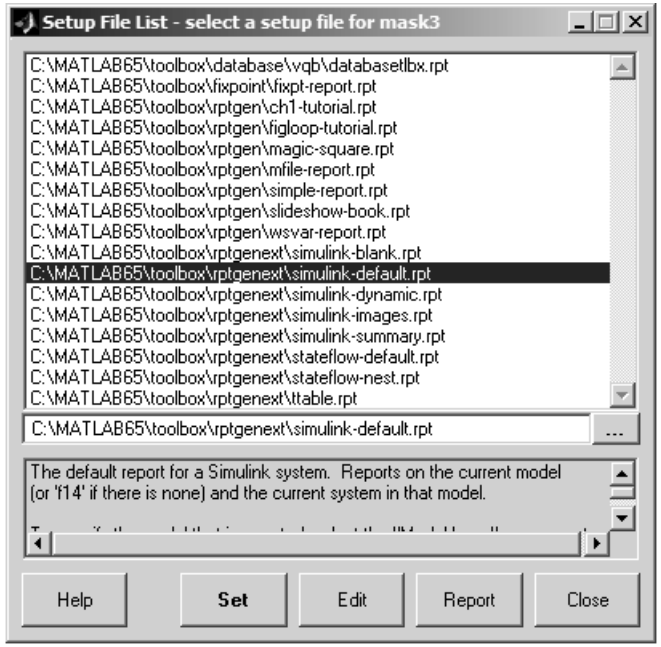


Рис. 9.12. Окно генератора отчетов

Уже из начала отчета видно, что он составлен в лучших традициях HTML-документов. Каждый раздел отчета представлен гиперссылкой, 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), тогда как генератор отчетов Simulink имеет множество других форм – см. рис. 9.12.

Рисунок 9.15 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. 9.16 показано продолжение описания модели. Здесь приводятся графическое изображение модели и спецификация ее блоков. Изображение модели точно такое же, как в исходной модели.

Наконец, на рис. 9.17 представлена заключительная часть отчета – осциллограммы модели.

Итак, полученный отчет отличается детальностью. Пожалуй, даже большей, чем в описании модели `mask3`, по которой готовился отчет. Его можно использовать как непосредственно в отчетных материалах по работе, так и для подготовки справочных материалов для библиотек пользователя моделей.

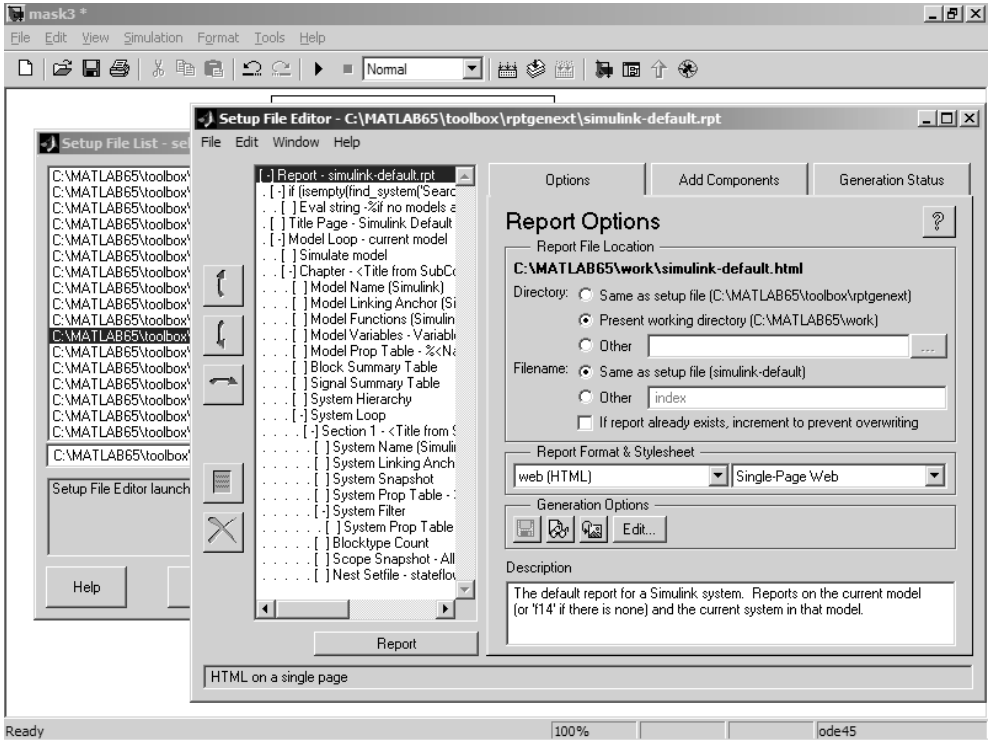


Рис. 9.13. Окно редактирования отчетов

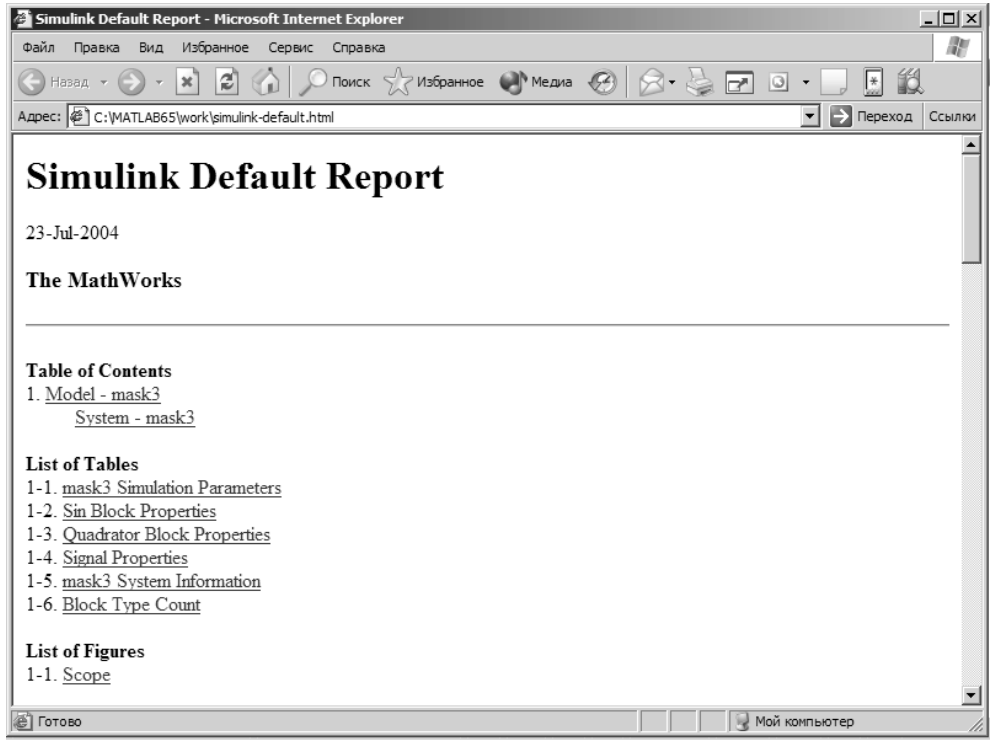


Рис. 9.14. Начало HTML-отчета по модели mask3

9.5. Инструменты ускорения моделирования

9.5.1. Профилировщик `Profiler`

Команда **Profiler** в позиции меню **Tools** включения/выключения профилирования моделей. Это мощное средство, обеспечивающее оценку скорости выполнения того или иного шага моделирования. По умолчанию данная опция выключена. Для включения профилировщика достаточно установить данную опцию включенной и пустить текущую модель на исполнение. Для нашей модели начало профилирования показано на рис. 9.18.

Нетрудно заметить, что при включенной опции профилирования исполнение модели ведет к генерации отчета о профилировании, который появляется в окне **Help** – на рис. 9.18 оно показано справа от модели.

В верхней части отчета видны четыре гиперссылки, относящиеся к разделам отчета:

- `Summary` – суммарный отчет;
- `Function Details` – детали работы функций;
- `Simulink Profile Help` – справка по профилометру;
- `Clear Highlighted` – сброс подсветки блоков.

Уже суммарный отчет дает достаточно полное представление о факторах, влияющих на время моделирования. На рис. 9.19 представлена почти полная таблица этого отчета по очень простой модели mask3. Нетрудно заметить, что время выполнения даже повторяющихся операций с простыми блоками арифметических операций очень мало и индицируется (с учетом ограниченной разрядности результатов вычислений) как нулевое. Заметим, что это характерно для вполне современного ПК, на котором выполнено моделирование, – это компьютер с процессором Pentium IV HT с частотой 2,6 ГГц.

Рисунок 9.20 показывает начало раздела по деталям функций. Нетрудно подметить, что данной простой модели этот раздел ничего особо нового не добавляет. Однако в случае более сложных моделей, в том числе демонстрационных из раздела **Demos** справки, ситуация может существенно измениться.

Остальные разделы отчета по профилированию в особых комментариях не нуждаются. В профилировании нуждаются практически все модели, которые создаются профессионально и могут многократно использоваться.

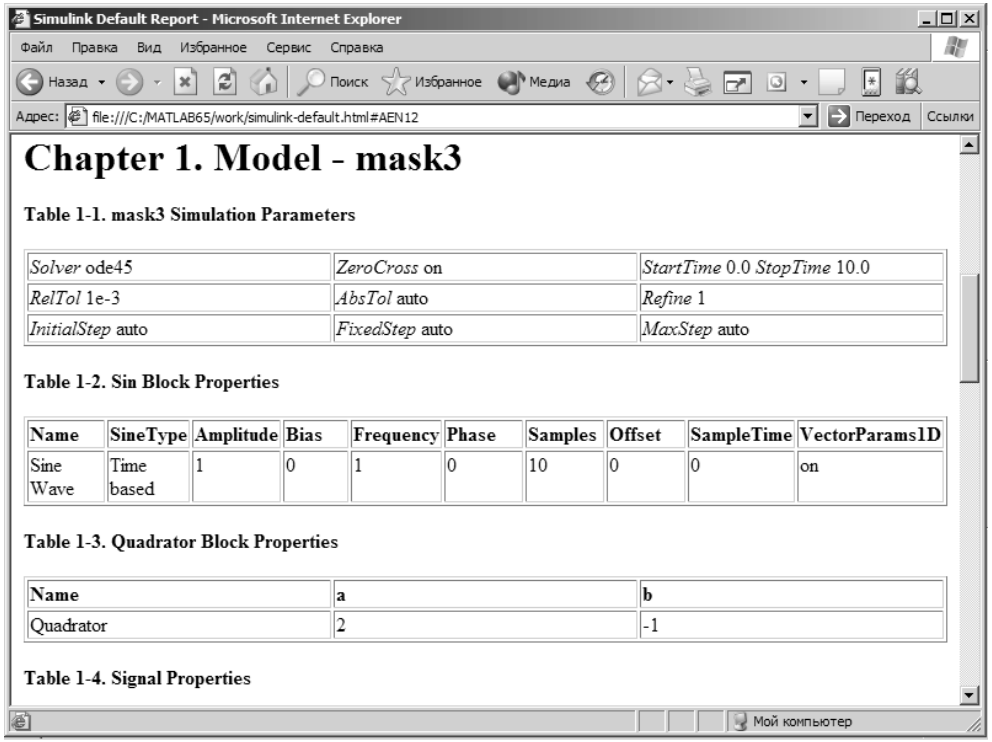


Рис. 9.15. Описание модели и ее блоков в отчете

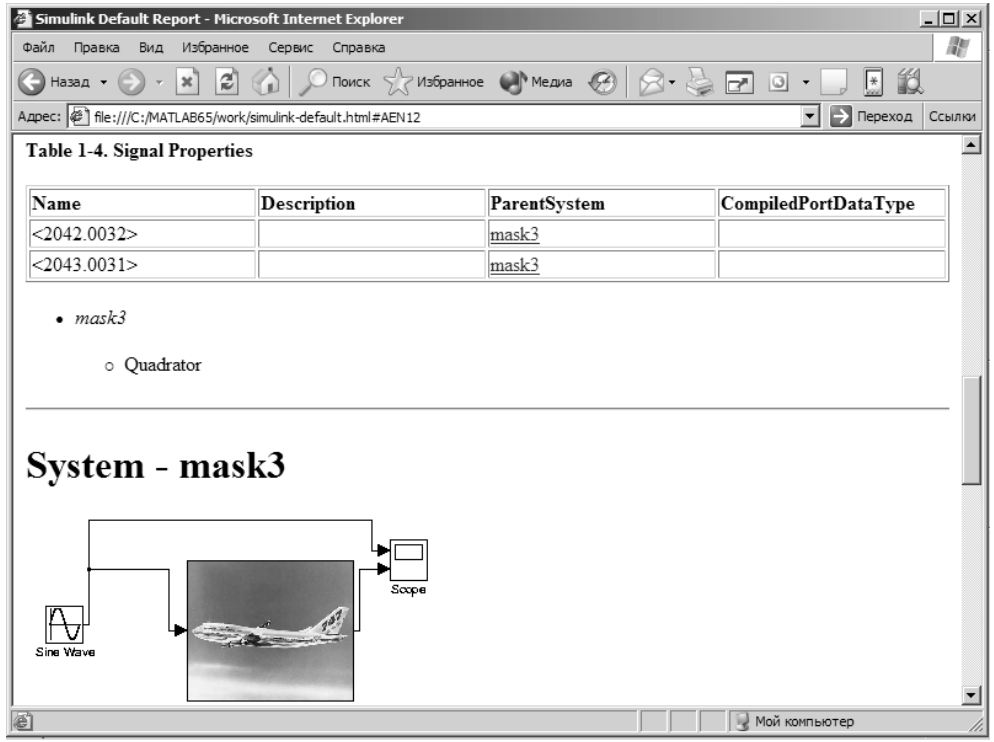


Рис. 9.16. Изображение модели и спецификация блоков в отчете

9.5.2. Применение Simulink-ускорителя

Скорость выполнения простых и демонстрационных моделей обманчиво велика, и некоторые из них выполняются на современных ПК в секунды, а то и доли секунды. Однако при решении серьезных задач моделирования она может быть неудовлетворительной, даже на самых мощных компьютерах. Нередко время моделирования в Simulink достигает нескольких часов, а то и суток. Поэтому важно знать меры по ускорению моделирования. В фирменном описании [96] отмечаются следующие приемы повышения скорости моделирования:

- применение Simulink-ускорителя (акселератора);
- переход от непрерывных моделей к дискретным;
- рациональный выбор метода и шага интегрирования;
- устранение избыточной точности вычислений;
- сокращение интервала времени моделирования;
- отказ от моделей, описываемых жесткими системами дифференциальных уравнений;
- исключение из моделей алгебраических контуров;
- выравнивание тактов дискретности;
- ограничение числа осциллографов;
- ограничение применения MATLAB-средств, таких как блоки `MATLAB Fcn` и блоки S-функций;
- исключение из моделей блоков памяти `Memory`, заставляющих решатели ОДУ переходить от высокоэффективных методов решения ДУ к методам первого порядка;
- ограничение применения блоков генерации случайных чисел Random Number.

Одним из действенных методов уменьшения времени моделирования является применение Simulink-ускорителя, или акселератора (Accelerator). В режиме моделирования Normal система Simulink осуществляет моделирование в интерпретирующем режиме. Переход в режим Accelerator означает компиляцию модели и создание быстро исполняемых кодов. Получающаяся при этом модель представляет собой файл динамической библиотеки *_acc.dll, где * – имя модели. При этом используются компоненты системы работы в реальном времени Real-Time Workshop и C-компилятор lcc, включенный в поставку Simulink.

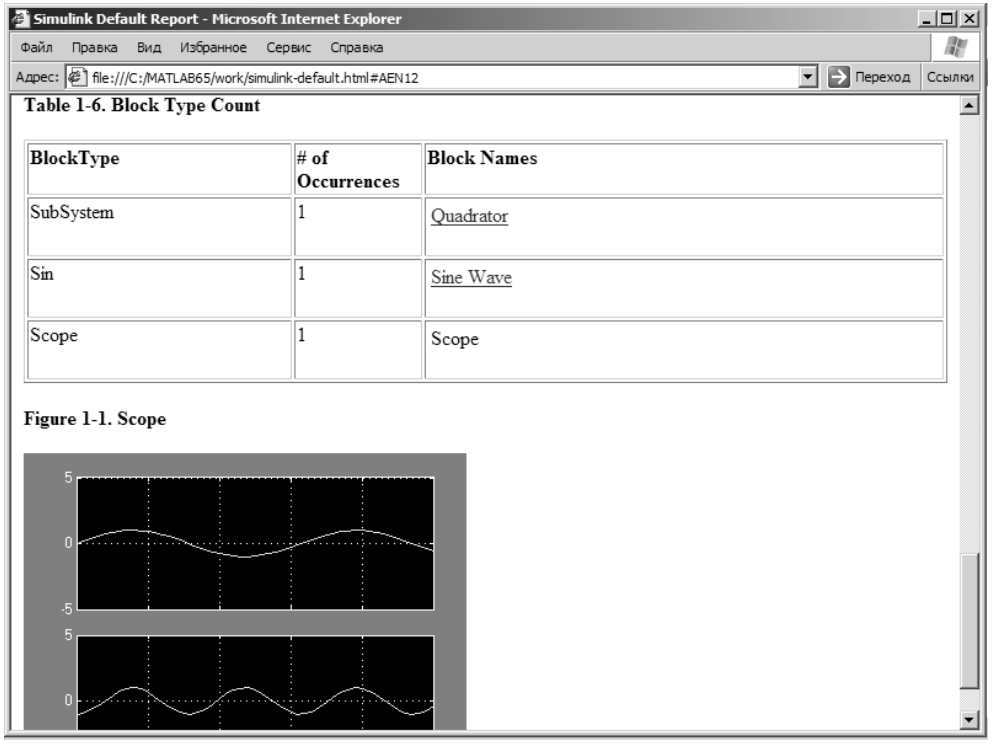


Рис. 9.17. Осциллограммы модели в отчете

Для включения Simulink-акселератора достаточно поставить мышью птичку у опции **Accelerator** в позиции **Simulation** меню. После этого, пустив модель на исполнение, в окне **MATLAB** можно наблюдать ход компиляции модели, а затем увидеть и ее исполнение.

Попробуем оценить на практике выигрыш в скорости моделирования при применении Simulink-ускорителя. Для этого запустим модель f14 на моделирование в режиме Normal, с включенной опцией создания отчета. Будет выведен отчет, который показан на рис. 9.21. Обратите внимание на время моделирования – оно составило 3,81 с.

На рис. 9.22 показана информация по времени выполнения функций для модели f14. Здесь уже хорошо видна разница во времени выполнения различных блоков.

Теперь сменим режим моделирования с Normal на Accelerator и вновь пустим моделирование с профилированием. Новый результат представлен на рис. 9.23. На этот раз время моделирования равно 0, 42 с. Таким образом, оно уменьшилось почти в 10 раз, что является прекрасным примером эффективности Simulink-ускорителя.

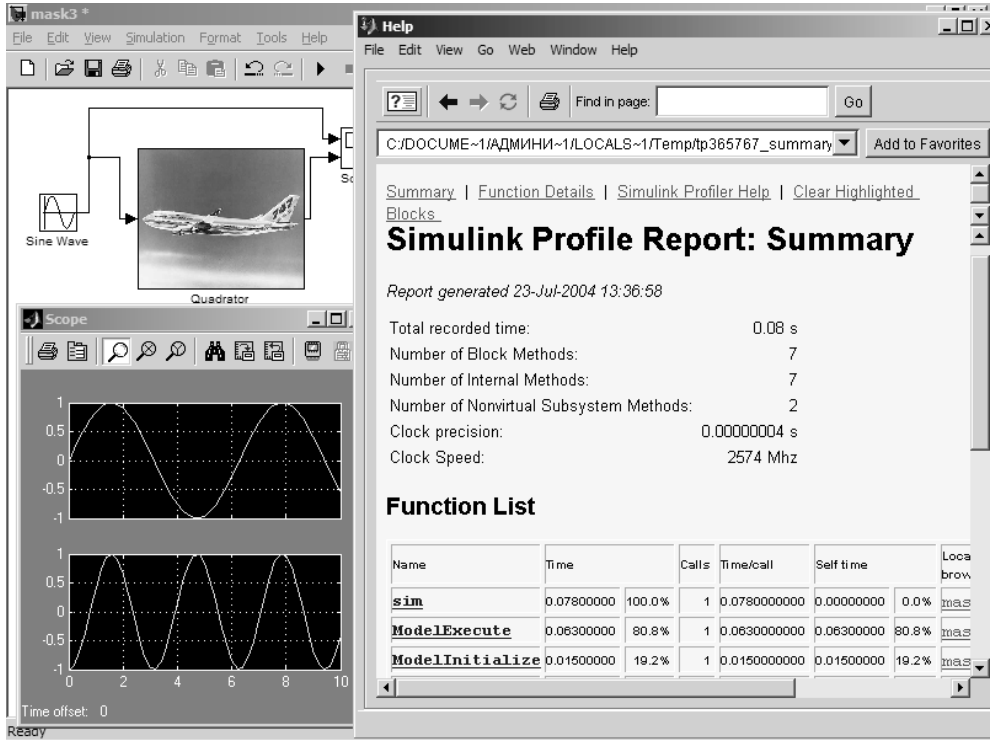


Рис. 9.18. Начало профилирования модели mask3

Можно вычислить примерное время моделирования и из окна MATLAB:

```
>> tic,[t,x,y]=sim ("f14",1000);toc
    elapsed_time =
        0.6880
```

Это время, ввиду иных условий запуска, несколько отличается от полученного с помощью профилирования.

9.5.3. Дискретизация моделей

Новым средством Simulink 5 является дискретизатор моделей. Он вводится исполнением команд **Model Descritizer...** в позиции **Tools** меню. Она открывает окно дискретизатора – см. пример на рис. 9.24.

В этом окне есть возможность просмотреть модель и выбрать блоки, которые можно сделать дискретными. Например, в нашем случае это блок генерации синусоидального сигнала. Рисунок 9.24 демонстрирует работу модели после дискретизации этого блока. Дискретизация отчетливо видна как ступенчатая синусоида. После дискретизации модель может выполняться быстрее, порою во много раз.

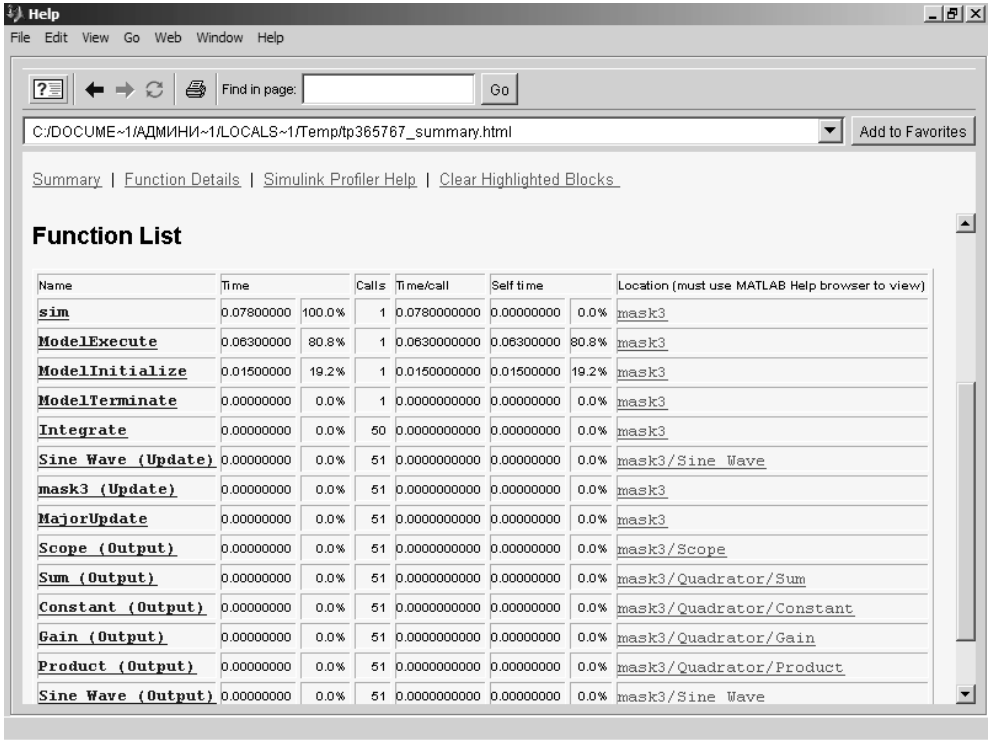


Рис. 9.19. Таблица из суммарного отчета профилирования модели mask3

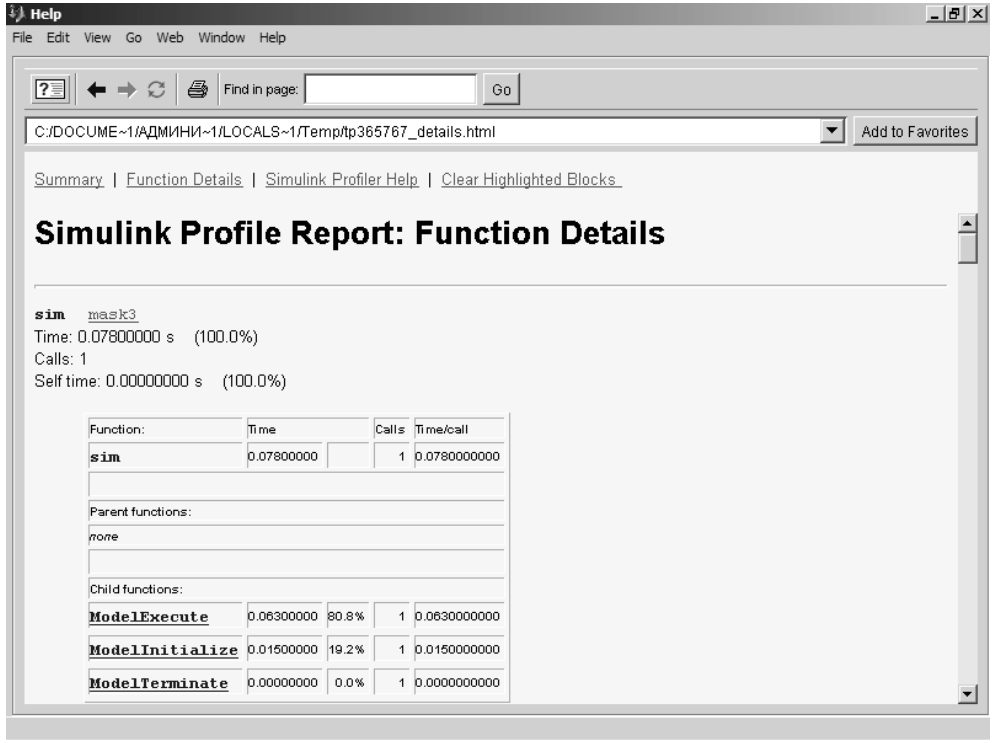


Рис. 9.20. Раздел `Function Details` *из отчета профилирования модели mask3*

9.6. Работа с LTI-вьювером

9.6.1. Вызов LTI-вьювера командой `Linear analysis`...

Команда `Linear analysis...` служит для открытия окна с результатами линейного анализа систем. Эта команда полезна для специальных моделей, в которых может быть реализован данный вид анализа. Модели могут быть как линейными, так и нелинейными. В последнем случае автоматически реализуется линеаризация выбранного фрагмента модели.

Рассмотрим типовую модель линейной системы с комплексной отрицательной обратной связью – рис. 9.25. Прямое и обратное звенья системы реализованы блоками операторной передаточной характеристики `Fcn`. Как видно из приведенных осциллограмм, реакция системы имеет колебательный характер, типичный для системы второго порядка.

Для проведения линейного анализа выполним команду `Linear analysis...` в позиции `Tools` меню. Появятся пустое окно **LTI-Viewer**, окно с предупреждающим сообщением и окно с портами входа и выхода вьювера. Эти окна показаны на рис. 9.26.

После этого необходимо перенести порты входа и выхода вьювера в модель и расположить их так, чтобы они выделяли нужный фрагмент – в нашем случае два звена с обратной связью. В окне вьювера надо выполнить команду **Get Linearized Model** – выполнить линеаризацию модели. В окне вьювера тут же появится график переходной характеристики системы – зависимости ее выходного сигнала от времени при наличии на входе единичного скачка. Это показано на рис. 9.27.

9.6.2. Выбор состояния системы

Отвлечемся от дальнейшего анализа системы и отметим одно важное обстоятельство – относительно какого состояния выполняется линеаризация. Это состояние можно менять с помощью окна рис. 9.28, выводимого командой **Set Operating Point...** в позиции **Simulink** окна LTI-вьювера. Заодно отметим, что в этой позиции меню есть и команда удаления портов вьювера **Remove Input/Output Points**.

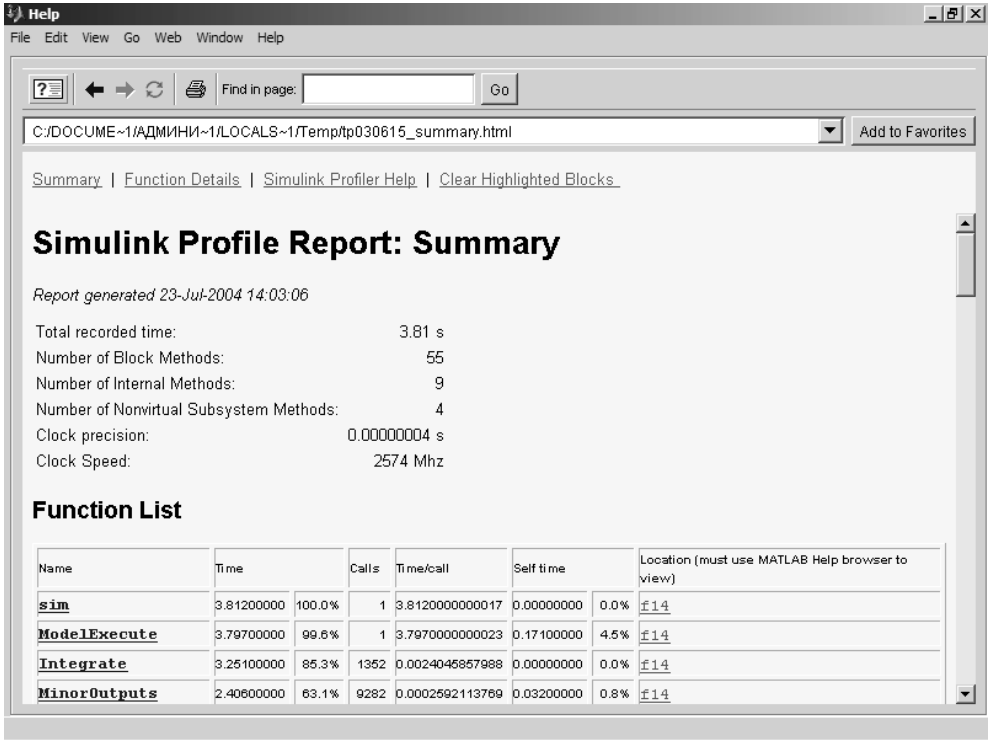


Рис. 9.21. Начало отчета по профилированию модели f14 без ускорения

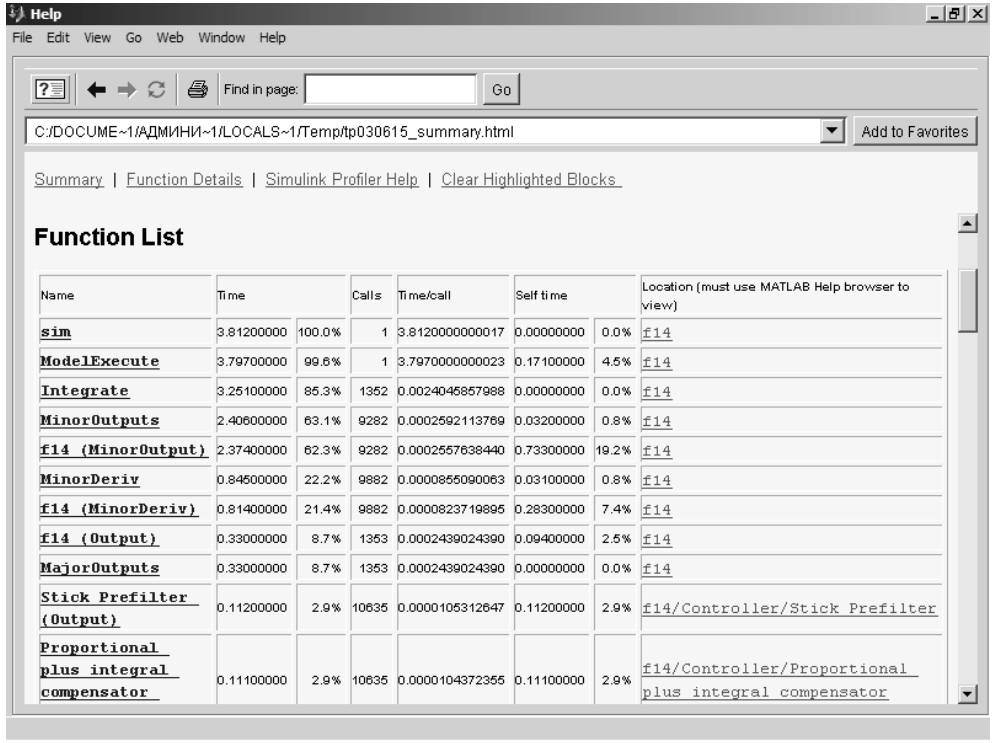


Рис. 9.22. Таблица времен выполнения функций (блоков), начало отчета по профилированию модели f14

В окне установки состояния системы можно установить одно из трех состояний для линеаризации:

- `Initial status in Simulink diagram` – соответствует исходному состоянию модели;
- `Zero status value` – состояние с нулевыми условиями;
- `User-defined state values` – состояние, задаваемое пользователем.

Последний выбор ведет к активизации списка имен блоков и панелей для ввода значений их начальных сигналов.

9.6.3. Выбор графических характеристик линейных систем

Но продолжим изучение возможностей LTI-вьювера. Он позволяет строить едва ли не все мыслимые характеристики линейных систем Response type:

- Step – переходная характеристика (реакция системы на единичный скачок);
- Impulse – импульсная характеристика (реакция системы на единичный импульс с площадью 1, бесконечно малой длительностью и бесконечно большой амплитудой);
- Bode – логарифмическая амплитудно-частотная и фазо-частотная характеристики;
- Bode Magnitude – только одна логарифмическая амплитудно-частотная характеристика;
- Nyquist – диаграмма Найквиста;
- Nichols – диаграмма Николса;
- Sigma – сингулярные числа системы;
- Pole/zero – нули/полюсы системы.

9.6.4. Конфигурация вывода графиков

Графики характеристик можно расположить в одном графическом окне, выбрав количество характеристик и вариант их расположения. Для этого достаточно исполнить команду конфигурации графиков **Plot Configurations...** в позиции **Edit**

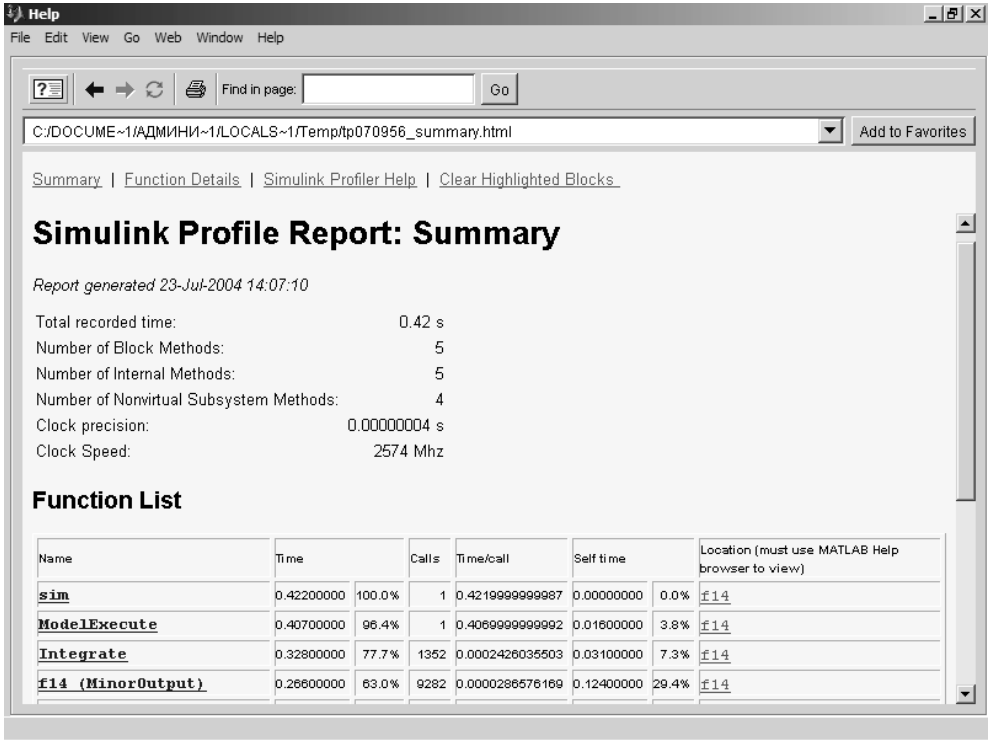


Рис. 9.23. Начало отчета по профилированию модели f14 с ускорением

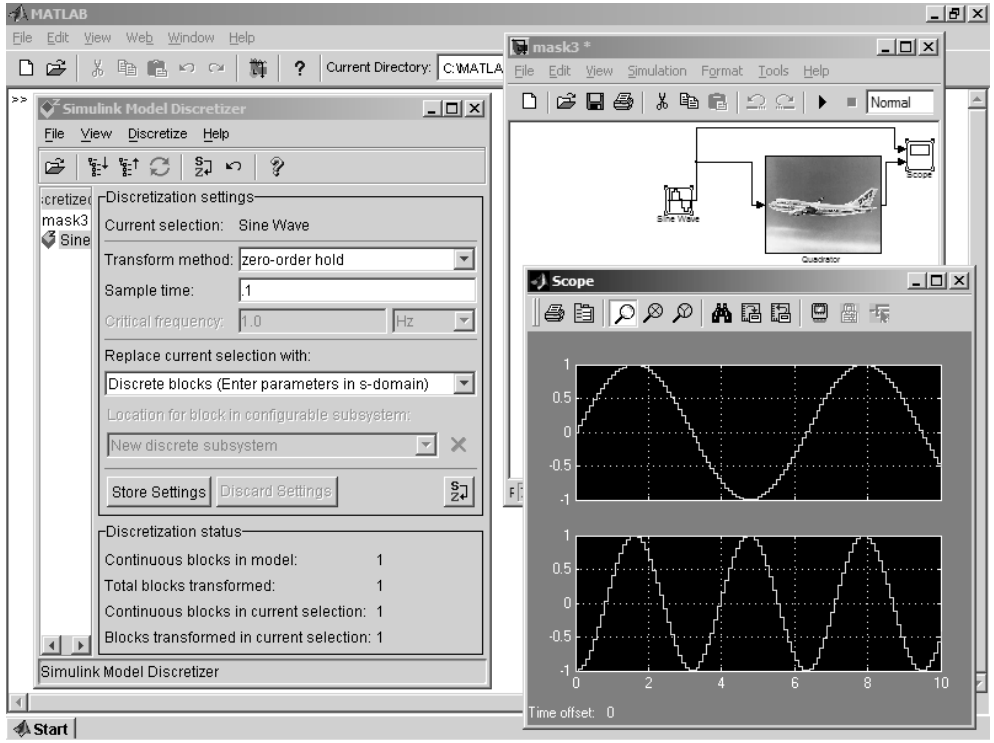


Рис. 9.24. Работа с окном дискретизатора

меню окна вьювера. Это приведет к появлению окна конфигурации, показанного на рис. 9.29. Для каждого подокна графики можно задать из списка одну из указанных выше характеристик.

Выберем, к примеру, конфигурацию из 4 графиков – рис. 9.29 с показанными справа списками характеристик. Тогда окно вьювера будет таким, как оно показано на рис. 9.30. Здесь построены переходная, импульсная, амплитудно-частотная и фазо-частотная характеристики системы и диаграмма Бодэ.

Рисунок 9.31 еще более информационный. Здесь, помимо уже отмеченных характеристик линейной системы, показаны еще две характеристики – диаграммы Найквиста и Николса. Они часто используются для графического анализа устойчивости линейных систем с обратными связями.

9.6.5. Пример линеаризации нелинейной системы

Как уже отмечалось, команда `Linear analysis...` осуществляет линеаризацию выбранной части даже нелинейной системы. Проверим такую возможность на практике. Для этого в нашу систему добавим явно нелинейный элемент – ограничитель (рис. 9.32). Настроив пороги ограничителя на 0 и 3, убедимся, что ограничитель действует и срезает верхушку выходного сигнала. Это отчетливо видно на осциллограмме выходного сигнала, представленной на рис. 9.32 справа.

Пустив LTI-вьювер, можно убедиться в том, что линеаризация прошла успешно, в результате чего получены переходная и импульсная характеристики системы – окно в левом нижнем углу. Однако переходная характеристика ограничения выброса не содержит. Это и говорит о том, что LTI-вьювер позволяет получить характеристики линеаризованной модели, даже если она нелинейна.

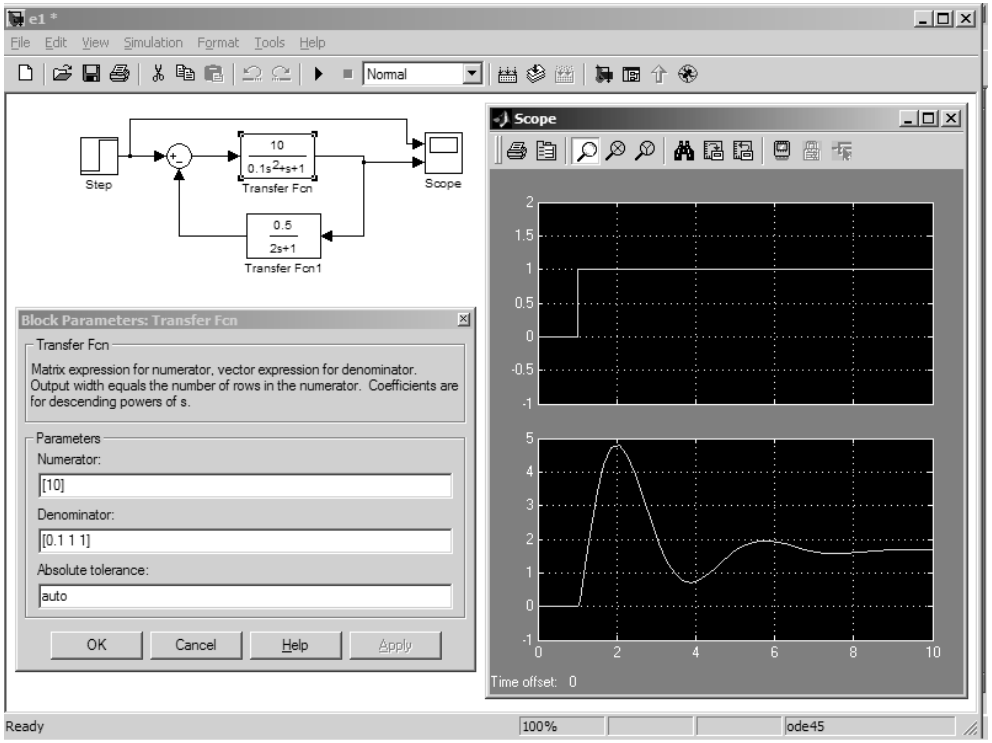


Рис. 9.25. Модель линейной системы с обратной связью

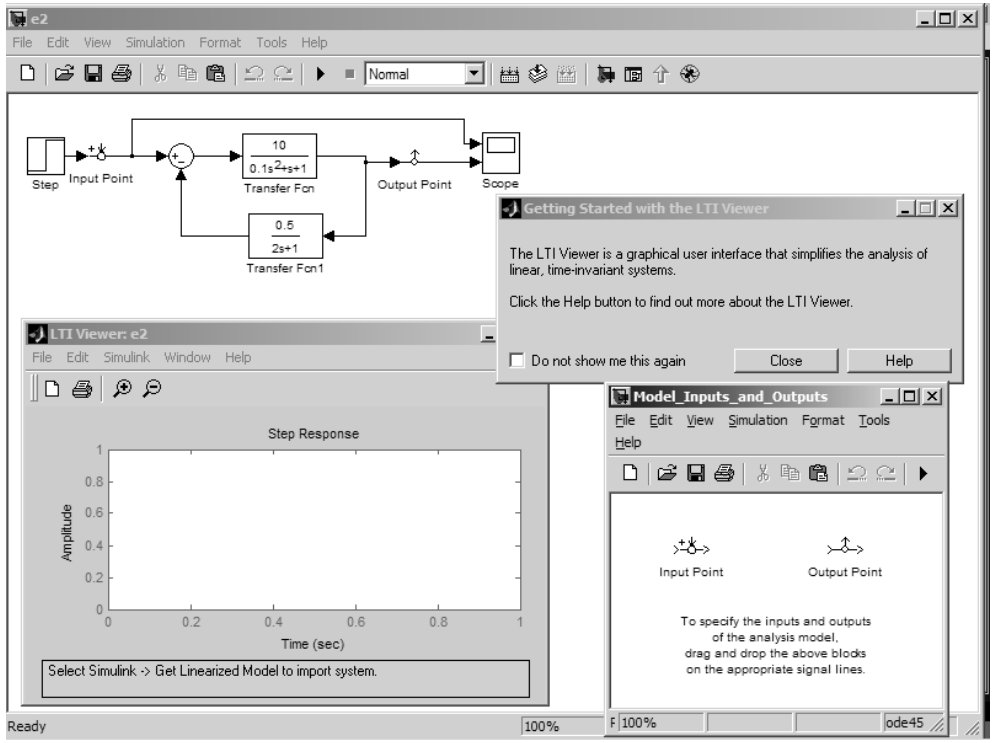


Рис. 9.26. Средства линейного анализа

9.7. Повышение эффективности и качества моделирования

9.7.1. Дополнительные средства в позиции Tools меню

В позиции **Tools** имеется и ряд других инструментальных средств.

- Команда **Real-Time Workshop** открывает доступ к так называемой мастерской реального времени. 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 setting...** открывает окно для установки параметров Simulink, нужных для работы с расширениями Simulink, использующими средства быстрых вычислений с фиксированной точкой.
- Команда **Requirements managements interface...** открывает окно интерфейсного навигатора **Requirements managements interface Navigator**.
- Команда `Look-up Table table editor` открывает окно редактора таблиц, позволяющего создавать сложные таблицы.
- Команда `Simulink Class Data Designer` открывает окно классов данных.

Эти команды носят явно специфический характер. Большинство пользователей системой Simulink ими едва ли воспользуются, а те, кому это понадобится, наверняка легко разберутся с ними самостоятельно.

9.7.2. Повышение скорости моделирования

Моделирование даже простых систем и устройств происходит порою достаточно медленно. Например, моделирование преобразовательных устройств с импульсным методом регулирования нередко требует времени моделирования порядка

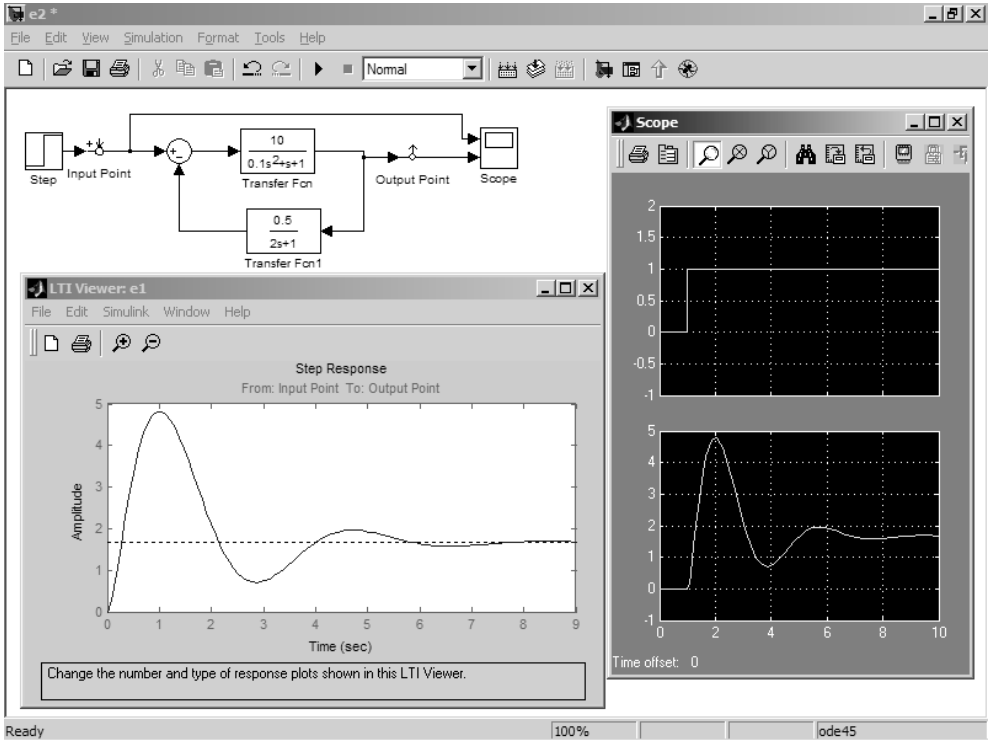


Рис. 9.27. Результаты моделирования линейной системы

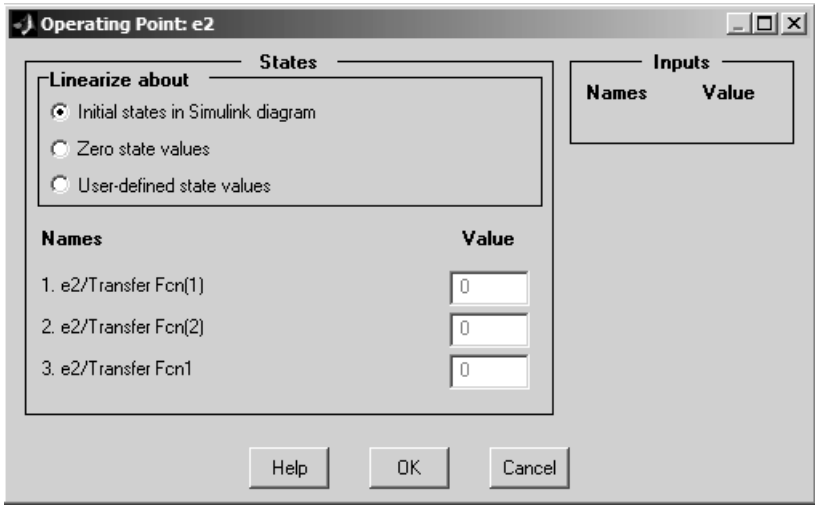


Рис. 9.28. Окно установки состояния системы при линеаризации

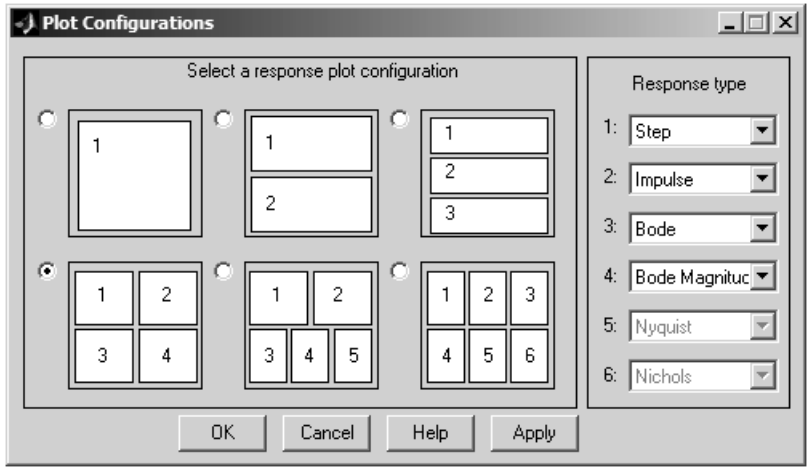


Рис. 9.29. Окно установки конфигурации графиков линейной системы

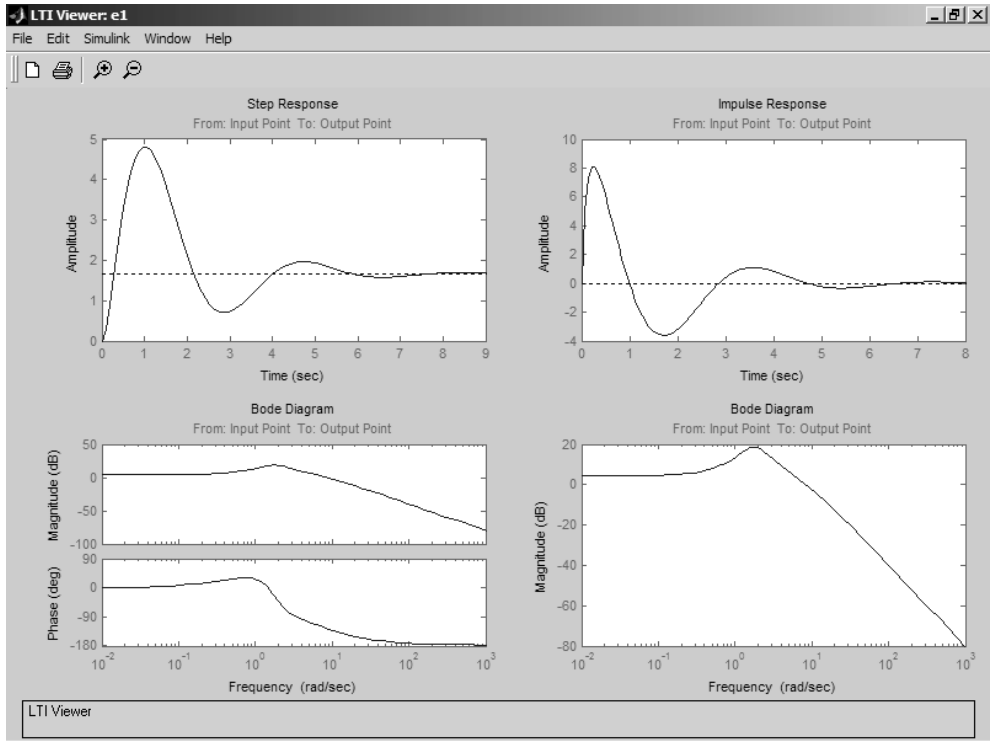


Рис. 9.30. Переходная, импульсная, амплитудно-частотная и фазо-частотная характеристики системы и диаграмма Бодэ

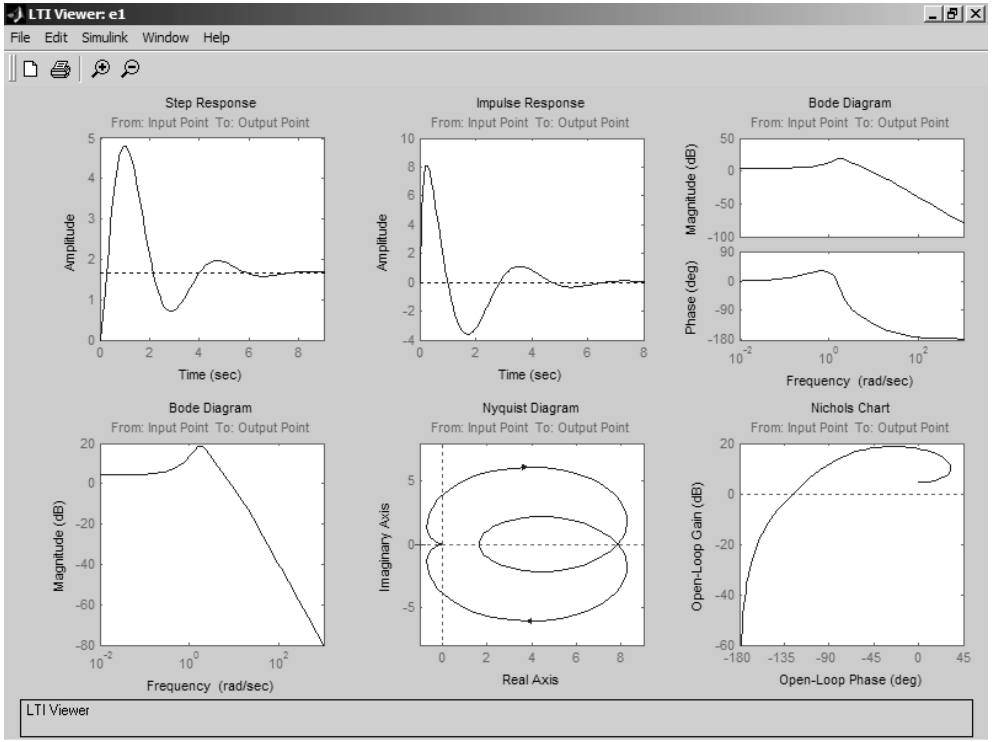


Рис. 9.31. Шесть характеристик линейной системы на одном графике

нескольких часов. В связи с этим важно знать основные способы повышения эффективности моделирования и прежде всего его ускорения.

Основные рекомендации по ускорению моделирования сводятся к следующим положениям.

- Применяйте для моделирования самые современные ПК, например с многоядерными микропроцессорами и с жесткими дисками, имеющими минимальное время доступа.
- Постарайтесь ограничить число программ, одновременно работающих с системой MATLAB + Simulink.
- Не мелочитесь при составлении моделей и применяйте по возможности укрупненные модели систем и устройств.
- Заменяйте по мере возможности непрерывные модели на дискретные, которые обычно моделируются заметно быстрее.
- Применяйте, где это возможно, ускоритель (акселератор) Simulink. Он создает исполняемые файлы типа .dll и способен ускорить моделирование в несколько раз. Это, однако, возможно, если диаграммы (модели) не имеют алгебраических контуров. Кроме того, при изменении структуры модели компиляция ее ускорителем повторяется заново.

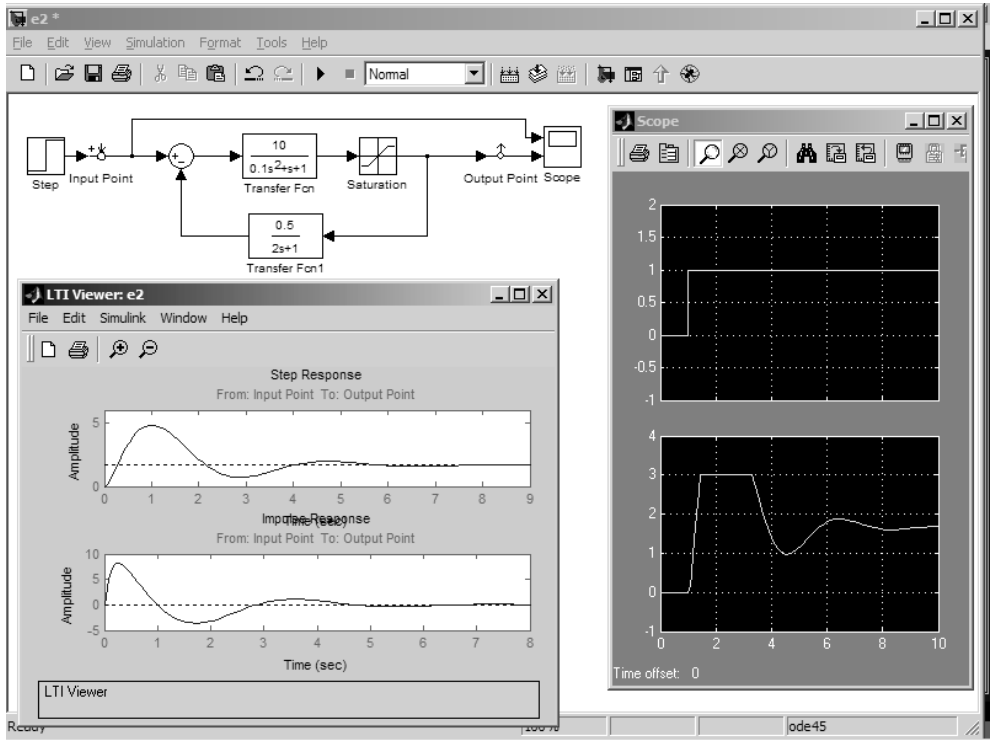


Рис. 9.32. Пример линеаризации нелинейной системы

- Ограничивайте значение максимального шага интегрирования Max step size. По умолчанию задано его значение auto, что может привести к резкому возрастанию числа шагов интегрирования и росту времени моделирования.
- Не увлекайтесь повышением точности моделирования. Почти всегда повышение точности (снижение погрешности) ведет к замедлению процесса моделирования. Для большинства задач численного моделирования достаточна относительная погрешность порядка 0,1% (задается параметром моделирования Relative tolerance).
- Ограничивайте максимальное время моделирования. У некоторых примеров оно задано равным бесконечности, и моделирование будет неограниченно долгим. Максимальное время моделирования надо выбирать таким, чтобы после выхода переходных процессов в стационарное состояние моделирование было завершено.
- Исключайте, если это возможно, алгебраические контуры в диаграмме, поскольку при их наличии на каждом шаге интегрирования используются медленные итерационные методы.

- Если модель имеет очень большой разброс постоянных времени, то применяйте специальные решатели жестких систем дифференциальных уравнений, например ode15s и ode23tb.
- Ограничивайте число индикаторов осциллографического типа, поскольку построение осциллограмм – довольно сложный и длительный процесс. Не стоит задавать число точек осциллограмм больше числа шагов моделирования.
- Откажитесь от применения блоков MATLAB Fcn, исполнение которых основано на вызове М-функций, выполняемых MATLAB в режиме интерпретатора.
- Не применяйте S-функции на языке MATLAB, поскольку они выполняются медленно. Вместо этого применяйте S-функции, созданные с помощью компиляторов C, Fortran и др.
- Примеряйте профилометр для контроля времени исполнения основных операций моделирования и старайтесь уменьшить затраты времени на исполнение каждого шага. Разумеется, это нужно, если отлаживается достаточно сложная модель, которая рассчитана на многократное применение в дальнейшем.

Учет этих рекомендаций позволяет значительно повысить эффективность и скорость моделирования даже относительно простых систем и устройств. Определенные трудности имеет моделирование систем с обратными связями – оно нередко требует применения методов интегрирования с переменным шагом и малой погрешностью интегрирования. Самую высокую эффективность моделирования обеспечивает правильный выбор модели устройств и систем, основанный на тщательном учете возможных алгоритмов моделирования. Впрочем, это относится уже не столько к технике, сколько к искусству моделирования и достигается многолетним опытом работы в данном направлении.

9.8. Практические примеры моделирования

9.8.1. Построение спирали Карно

Спираль Карно – хороший пример на применение интеграторов расширения Simulink. Спираль строится с применением (при изменении t от 0 до $t_{макс}$) следующих выражений:

$$x(t) = \int_0^t \cos(x^2)dt; \qquad y(t) = \int_0^t \sin(x^2)dt.$$

Они задают построение графика спирали Карно в параметрическом виде. Для этого используется виртуальный графопостроитель – рис. 9.33. Текущее время формируется генератором нарастающего напряжения, которое начинается с уровня –10 для построения двух ветвей спирали. Затем этот сигнал возводится

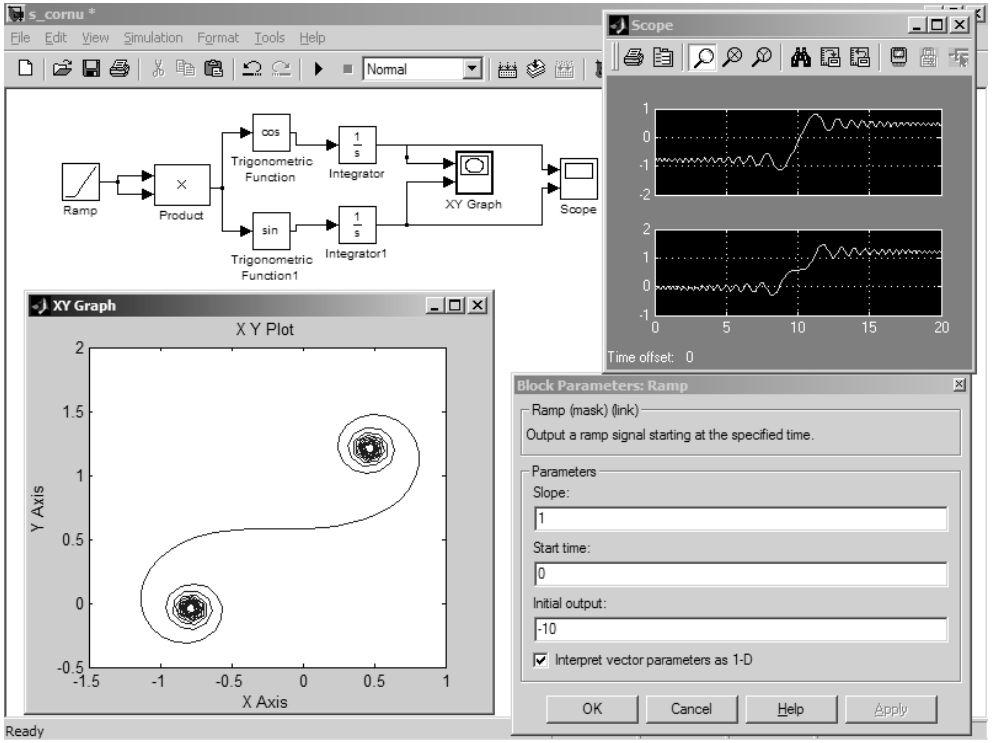


Рис. 9.33. Построение спирали Карно

в квадрат, и его синус и косинус интегрируются с помощью блока интеграторов. Осциллографы показывают временные зависимости сигналов на входах интеграторов.

9.8.2. Синтез АМ-сигнала

Радиотехники знают, что амплитудно-модулированный синусоидой гармонический сигнал имеет несущую частоту и боковые частоты. Проверим это суммированием сигнала с частотами 14, 15 (несущая частота) и 16 рад/с и амплитудами несущей 1 и 0,5 для боковых частот (это соответствует 100% модуляции). Рисунок 9.34 показывает справедливость данного утверждения – действительно суммирование трех указанных частот формирует АМ-колебание, представленное осциллограммой. Частота модулирующего сигнала при этом равна 1.

Рисунок 9.34 демонстрирует также применение блока спектрального анализа для получения спектра синтезированного АМ-колебания. Полученный спектр в виде АЧХ и ФЧХ сигнала представлен на рис. 9.35.

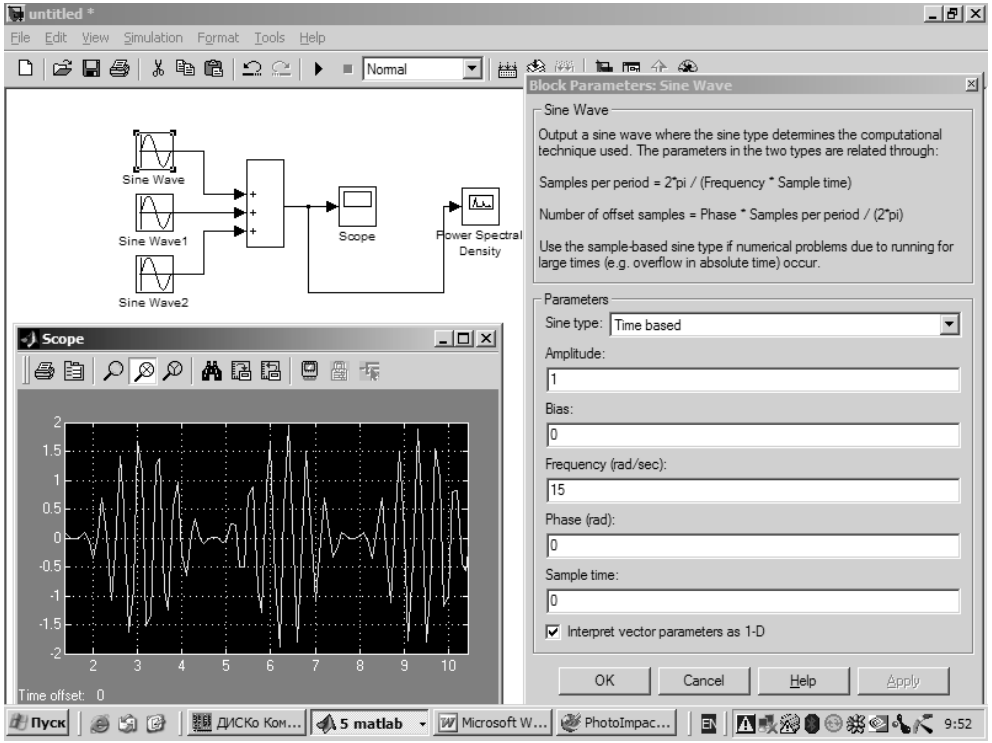


Рис. 9.34. Синтез АМ-колебания

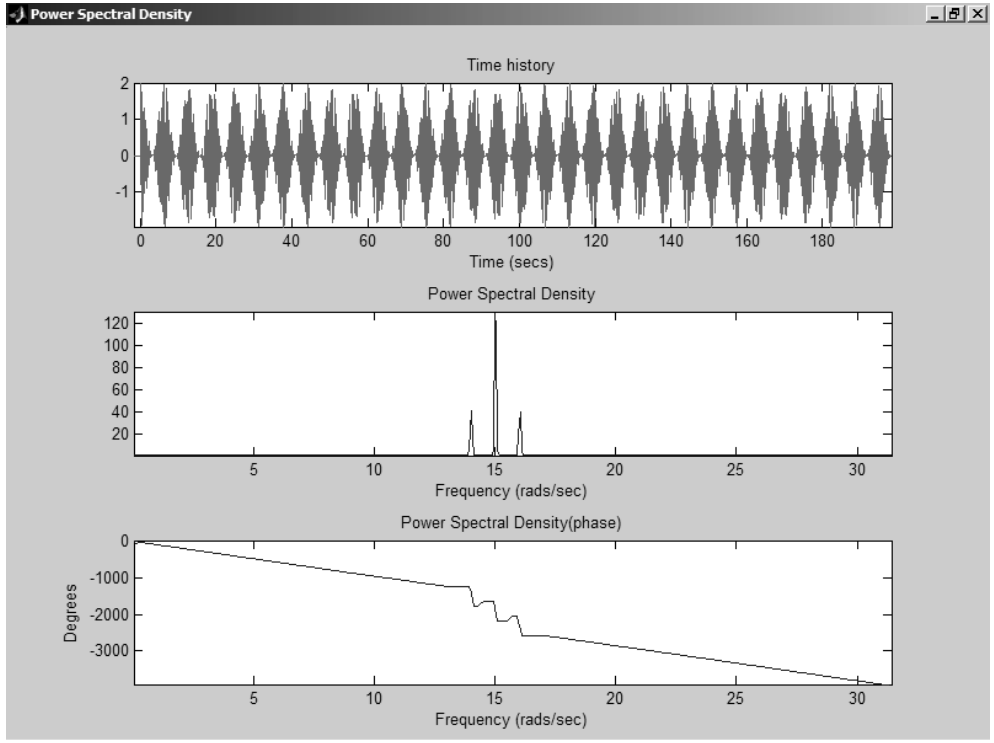


Рис. 9.35. Спектр синтезированного АМ-колебания

9.8.3. Нестабильные линейные системы с обратной связью

Если линейная система охвачена положительной или комплексной обратной связью, то она может вести себя как апериодическая или колебательная система, как стабильная, так и нестабильная. На рис. 9.36 показаны две явно нестабильные системы такого рода. Они представлены передаточными функциями Fcn, удобными для описания линейных систем, таких как электротехнические и радиотехнические цепи, фильтры, механические резонаторы и т. д.

Верхняя система ведет себя как апериодическая нестабильная система – сигнал на ее выходе экспоненциально возрастает и может достигать очень больших и на практике нереальных значений. Вторая система – колебательная, и сигнал на ее выходе имеет вид нарастающей синусоиды.

9.8.4. Получение незатухающих почти синусоидальных колебаний

Для получения незатухающих почти синусоидальных колебаний надо использовать колебательную нестабильную линейную систему, дополненную ограничителем. Это значит, что такая система становится нелинейной. Чтобы улучшить форму сигнала, можно расположить ограничитель на входе блока, задающего прямой канал системы с частотной избирательностью (в электронных генераторах эту роль обычно выполняет резонансный *LCR*-контур). Это иллюстрирует рис. 9.37, где сверху показана линейная система, а снизу – нелинейная.

Если на некоторой частоте колебаний существует баланс амплитуд и фаз, то колебания приобретают вид почти синусоидальных колебаний с этой частотой и с неизменной амплитудой. На этом и основаны генераторы синусоидальных колебаний.

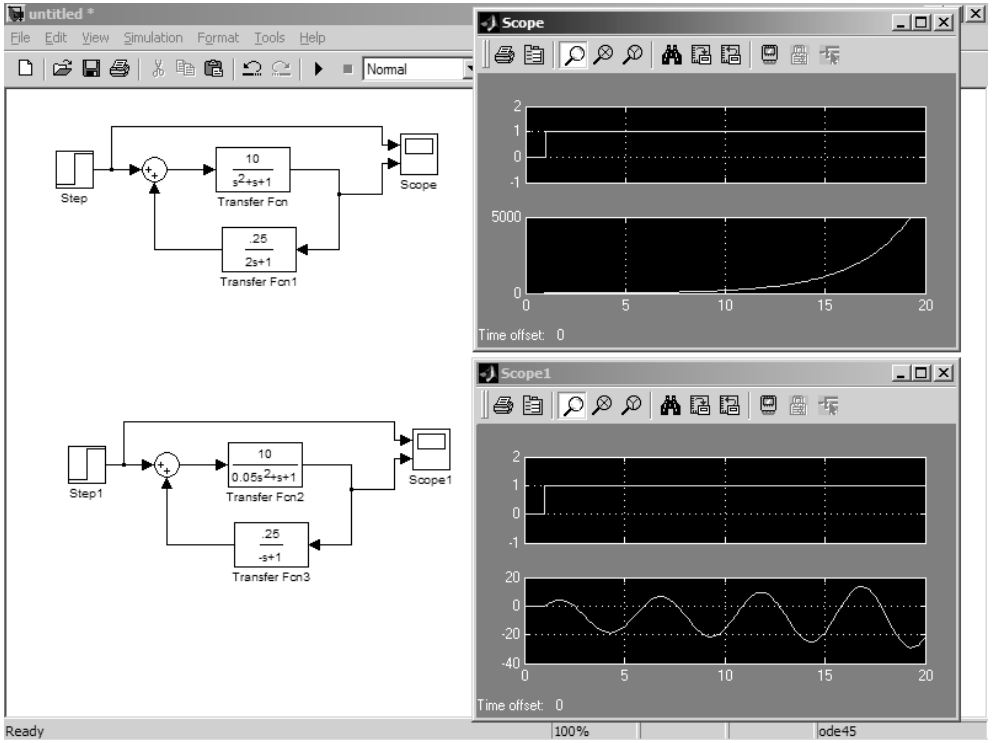


Рис. 9.36. Два примера нелинейных систем

Системы, в которых колебания возникают самопроизвольно и без каких-либо внешних воздействий, называют автономными. Представленные на рис. 9.37 системы не являются таковыми, поскольку в них используется источник внешнего воздействия в виде перепада.

Моделирование автономных систем может вызвать значительные трудности при определенном задании начальных условий. Поэтому приходится либо моделировать неявно заданные скачки какого-либо сигнала, либо имитировать шум, присущий любым реальным устройствам. Рисунок 9.38 иллюстрирует эти подходы.

В первой из систем источник постоянного сигнала на самом деле является источником перепада, действующим в момент t = 0 начала моделирования. Во втором случае имитируется шум на входе системы. Кроме того, во втором случае на входе блока прямого тракта имеется ограничитель, обеспечивающий установление режима колебаний с неизменной во времени амплитудой.

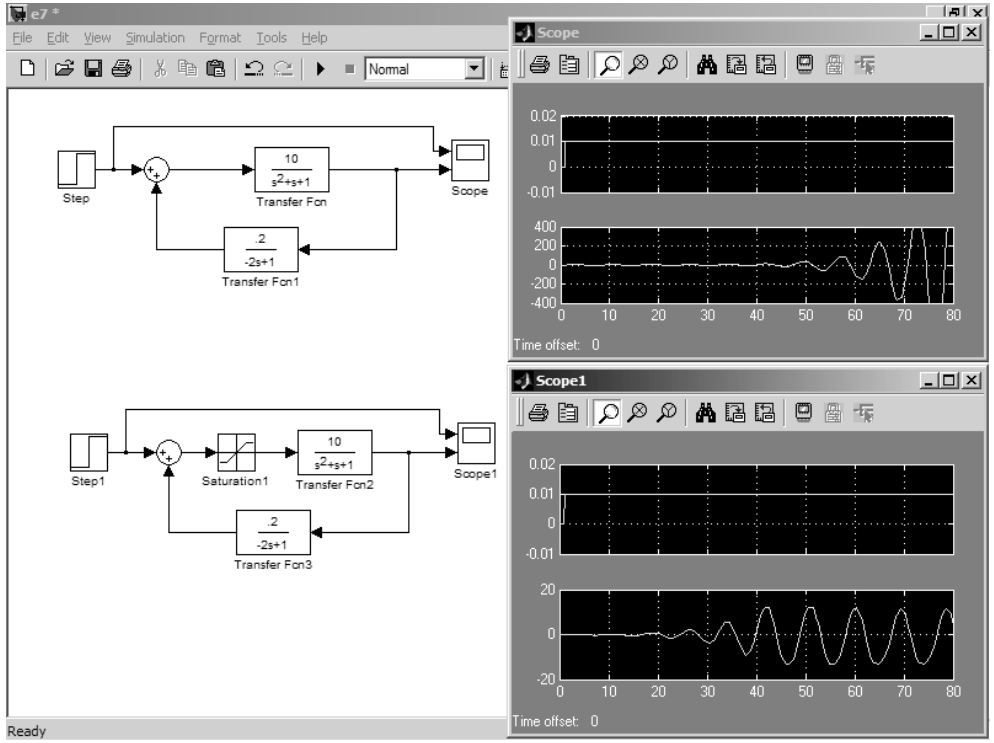


Рис. 9.37. Переход от линейной колебательной системы к нелинейной

9.9. Демонстрационные примеры Simulink

9.9.1. Доступ к демонстрационным примерам Simulink

В выполнении сложного и трудоемкого процесса математического моделирования важное значение имеет знакомство с профессионально подготовленными демонстрационными примерами. Доступ к демонстрационным примерам Simulink возможен из вкладки **Demos** справки системы MATLAB или Simulink. Рисунок 9.39 демонстрирует доступ к демонстрационным примерам Simulink.

В левой части окна справки видны четыре папки с демонстрационными примерами Simulink:

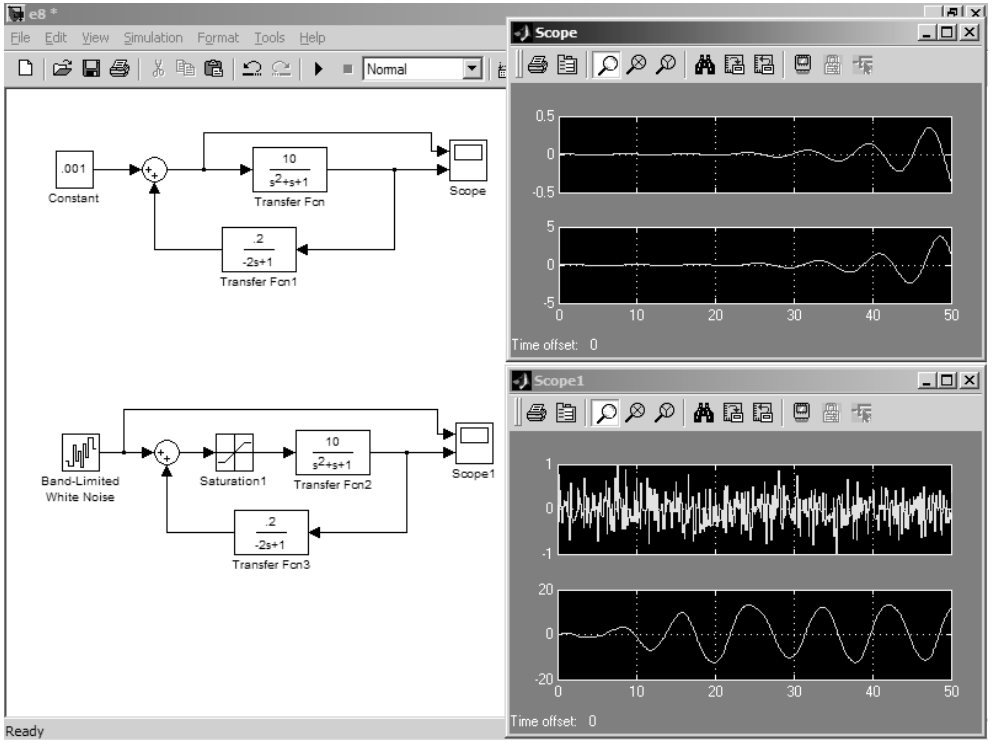


Рис. 9.38. Модели систем с «самопроизвольным» возникновением колебаний

- `Features` – особые примеры;
- `General` – примеры общего характера;
- `Automotive` – примеры моделирования транспортных и автомобильных систем;
- `Aerospace` – примеры моделирования систем летающих объектов.

Разумеется, примеры могут быть загружены в Simulink и обычным путем – из окна команды **Open**. Примеры легко идентифицировать по имени файла, которое видно в титульной строке окна Simulink.

9.9.2. Моделирование простого маятника

Динамические системы широко представлены в механике. Мы рассмотрим две такие системы. На рис. 9.40 представлена простая система механического маятника – подвешенного за верхний конец стержня. Маятник совершает сложные движения – наряду с быстрыми колебаниями на частоте резонанса заданы беспорядочные отклонения стержня от состояния равновесия.

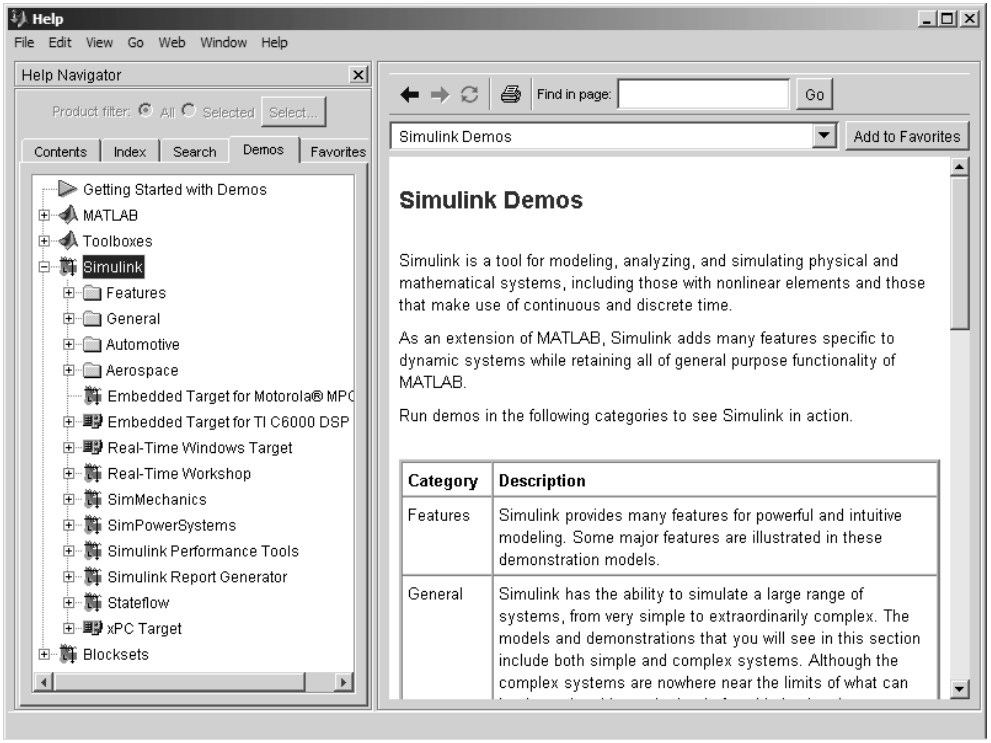


Рис. 9.39. Доступ к демонстрационным примерам Simulink

Модель маятника представлена инерционной системой с обратными связями. Блок **Animation Function** выводит графическое окно анимационного изображения колебаний. На нем можно видеть колебания стержня – маятника. На рис. 9.39 представлен лишь один кадр анимации.

9.9.3. Колебания многозвенного объекта

Гораздо сложнее оказывается поведение сложных многозвенных систем, со звеньями, имеющими степени свободы, находящихся под внешним воздействием. Рисунок 9.41 показывает модель колебаний объекта, состоящего из трех подвижно сочлененных стержней, находящихся под сложным воздействием. Каждое звено может поворачиваться относительно точки сочленения звеньев.

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

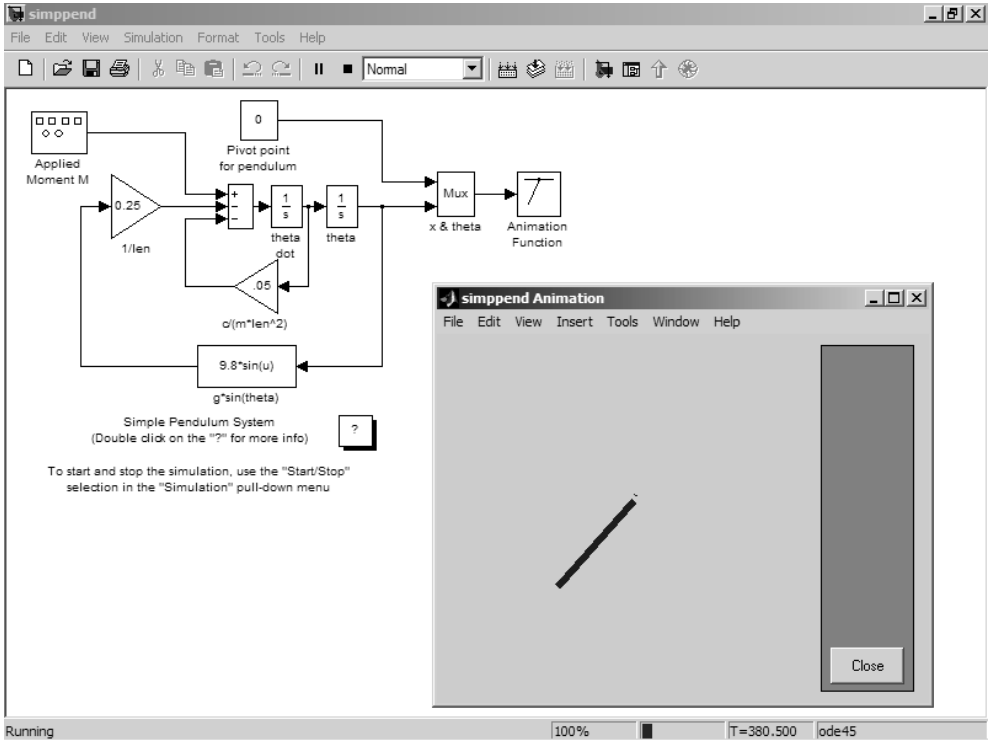


Рис. 9.40. Модель механического маятника

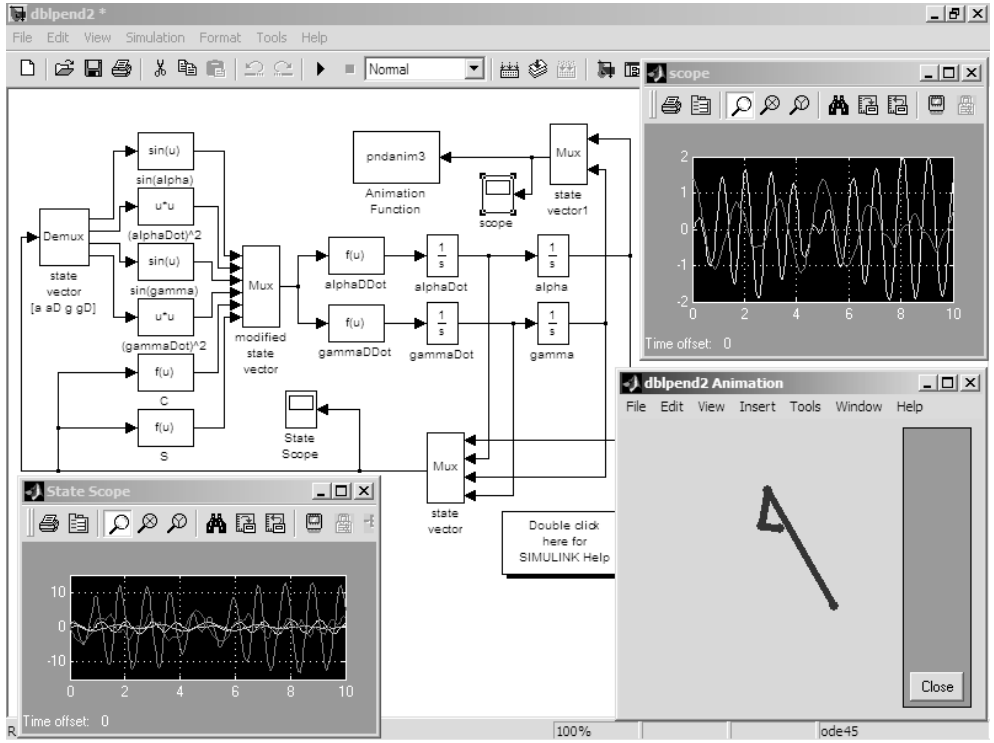


Рис. 9.41. Моделирование колебаний многозвенного объекта

9.9.4. Моделирование отскакивающего от поверхности мячика

Моделирование отскакивающего от поверхности (пола) мячика – любимая задача на математическое моделирование в физике. Пример ее решения из справки по Simulink показан на рис. 9.42. Дифференциальные уравнения, описывающие динамику модели (кстати, как и в предыдущих примерах), скрыты в модели и для пытливого читателя могут послужить поводом для тщательной разборки модели. При моделировании учитывается потеря энергии при каждом отскоке мячика.

Осциллограммы на рис. 9.42 показывают изменение скорости движения (она положительна при подскоке мячика и отрицательна при его падении) и положения мячика. Это еще один пример релаксационной механической системы с резким изменением состояния в момент падения мячика на поверхность.

9.9.5. Моделирование автопилота с аналоговыми блоками

Одним из первых примеров успешного применения Simulink стало моделирование автопилота – устройства, применяемого на воздушных судах для выдерживания заданного курса. Сейчас для этого применяются цифровые и дискретные системы и даже средства нечеткой логики. Но первые автопилоты строились на аналоговых устройствах и по сути представляли собой системы с обратной связью, корректирующей положение воздушного судна. Рисунок 9.43 показывает упрощенную модель такого автопилота и демонстрирует ее работу.

Одна из важных задач в построении подобных систем заключается в минимизации времени отклонения воздушного судна от заданного автопилотом курса. Страшная катастрофа над небом Германии, когда столкнулись российский и американский самолеты, указывает на актуальность решения этой задачи. К сожале-

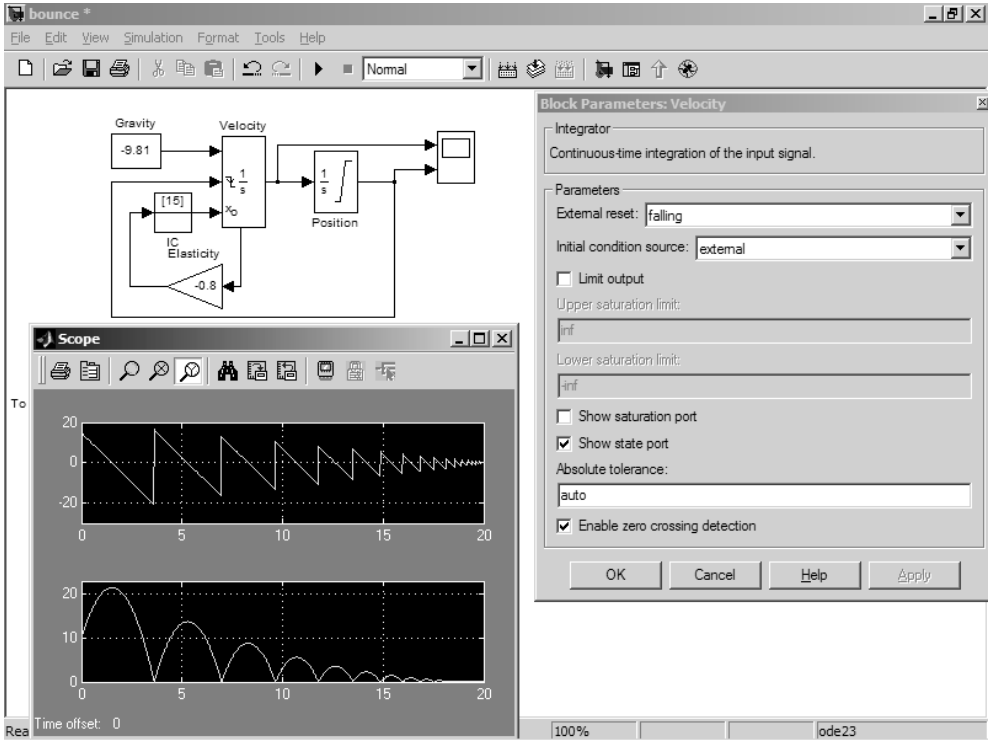


Рис. 9.42. Моделирование отскакивающего от поверхности мячика

нию, ускорение реакции автопилота (типичной сильно инерционной системы) на возникшее возмущение обычно сопровождается появлением характерного затухающего процесса, который ведет к рысканию воздушного судна при внезапной смене курса (именно этот случай и демонстрирует модель рис. 9.43).

Стремление уменьшить рыскание ведет к существенному усложнению автопилота, построенного на аналоговых блоках. В блок автопилота приходится вводить сложные многоконтурные обратные связи и корректирующие цепи. Усложненная модель автопилота показана на рис. 9.44. Она обеспечивает намного меньшие колебания воздушного судна при резкой смене курса.

9.9.6. Пример дискретной системы

Одним из путей усовершенствования систем с обратными связями, такими как автопилот, является переход к реализациям на дискретных компонентах и дискретных методах. Пример такой реализации из числа демонстрационных примеров Simulink 5/6 дан на рис. 9.45. Напомним, что дискретные блоки характеризуются передаточными функциями переменной z, которые могут использоваться для z-преобразований.

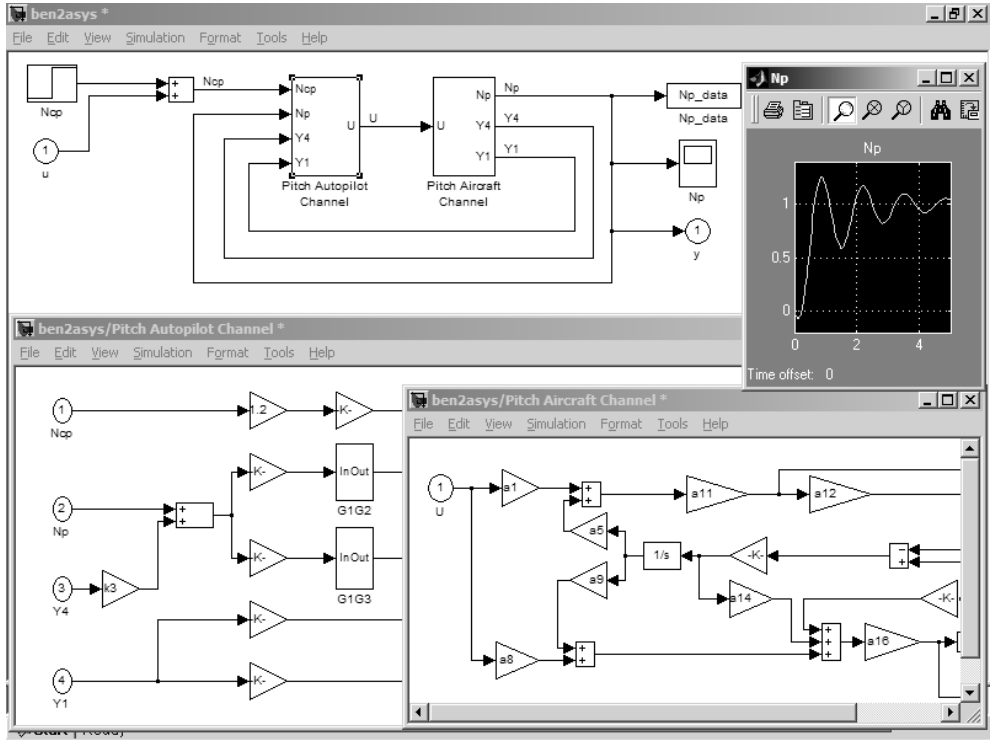


Рис. 9.43. Моделирование автопилота, построенного на аналоговых блоках

9.9.7. Применение примеров раздела Automotive

В разделе **Automotive** справки **Demos** имеется множество интересных примеров на применение систем, которые используются на движущемся транспорте, в частности автомобильном. Это системы стабилизации скорости и угла поворота, электрические и гидравлические системы управления и т. д. Из всего этого множества примеров рассмотрим первый – рис. 9.46.

Рисунок 9.46 показывает модель системы, управляемой дросселем. Это типичная ET-система, управляемая триггером.

9.9.8. Ранняя модель автопилота летательного аппарата F14

Раздел Aerospace – один из самых крупных разделов демонстрационных примеров. Система MATLAB + Simulink давно используется для моделирования авиа-

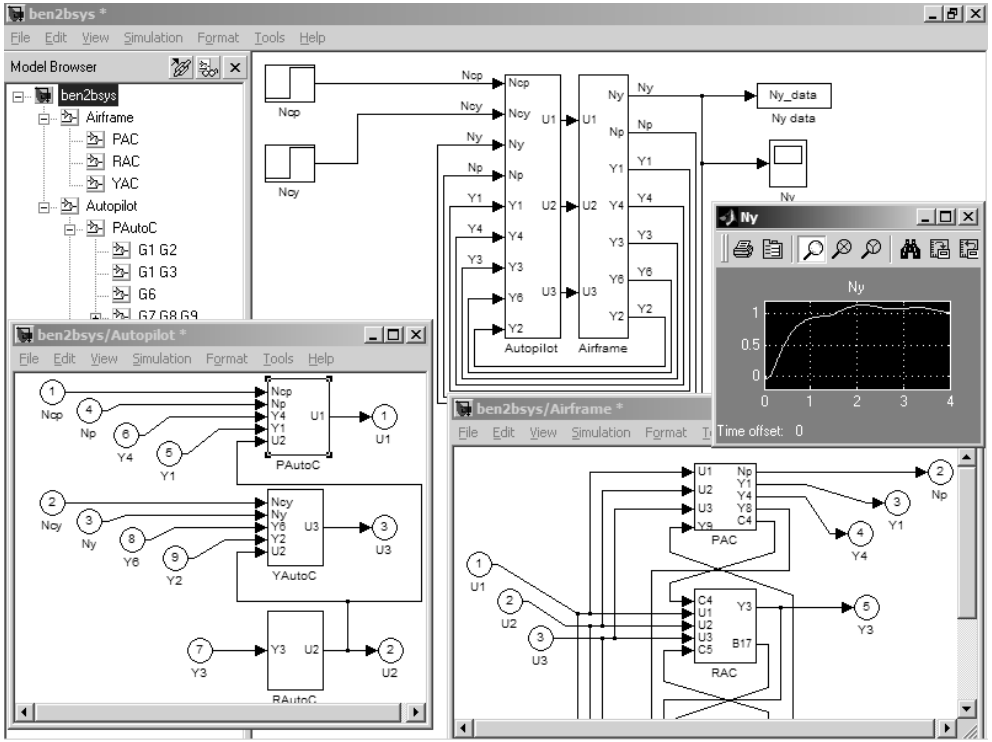


Рис. 9.44. Усложненная модель автопилота с малыми колебаниями реакции

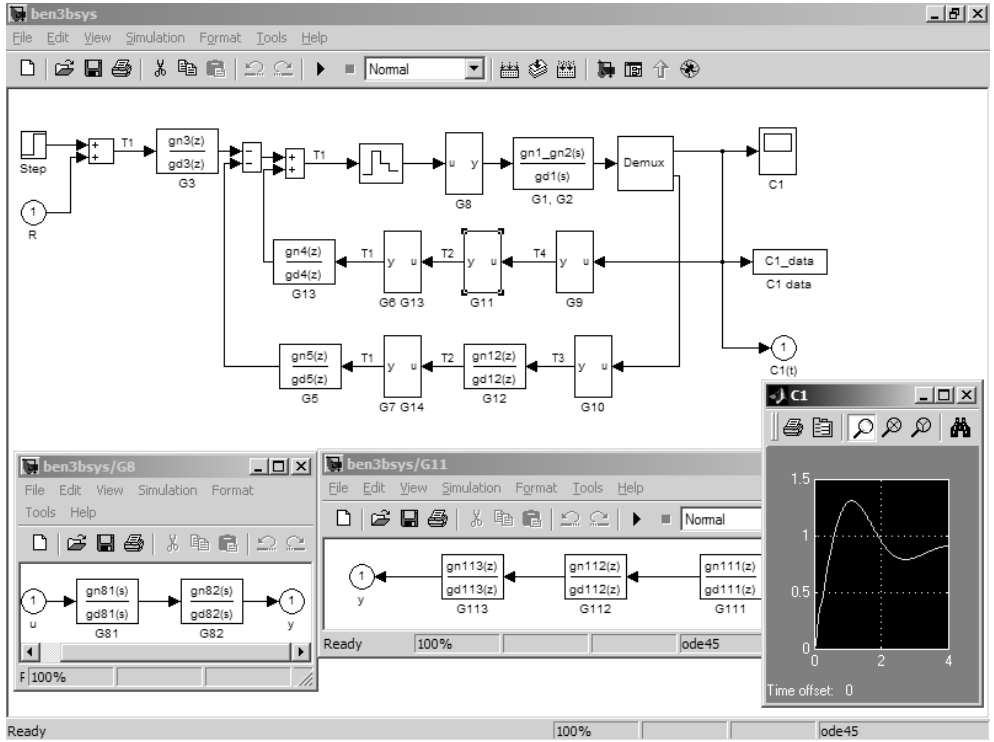


Рис. 9.45. Модель трехканальной системы с дискретными блоками

ционных систем и отдельных объектов. В разделе Aerospace можно найти множество примеров этого. Мы ограничимся рассмотрением только примеров, относящихся к системам, именуемым автопилотами. Рассмотрим модели, относящиеся к конкретному летательному аппарату – F14.

На рис. 9.47 представлена одна из ранних моделей упрощенной системы автопилота для аппарата F14. Модель описывает отработку летательным аппаратом скачка, задающего изменение курса. Работа модели контролируется тремя осциллографами, осциллограммы которых представлены на рис. 9.47. Можно отметить высокое качество переходных процессов – выбросы на осциллограммах едва заметны.

Модель выглядит простой для такой сложной системы, как автопилот. Но эта простота достигнута применением ряда подсистем, часть которых показана на рис. 9.48. Даже общий обзор этих подсистем показывает, что данная модель реализует аналоговые методы построения управляющих систем.

9.9.9. Комбинированная модель автопилота F14

На рис. 9.49 представлена демонстрационная комбинированная модель автопилота F14. Она имеет управляемые вручную (мышью) переключатели, позволяющие сравнивать аналоговую модель с двумя вариантами цифровых моделей. На рис. 9.49 переключатели задают аналоговую модель. Осциллограммы в правой части модели дают представление о ее работе в аналоговом режиме.

На другом рисунке (рис. 9.50) показана работа в одном из двух цифровых режимов. Можно подметить, что характеристики аналоговой и цифровой моделей близки, но цифровую модель всегда легче реализовать и контролировать.

Некоторые из подсистем данной модели представлены на рис. 9.51. В них, в свою очередь, входят другие подсистемы. Таким образом, данная модель является хорошим примером построения моделей с вложенными друг в друга подсистемами.

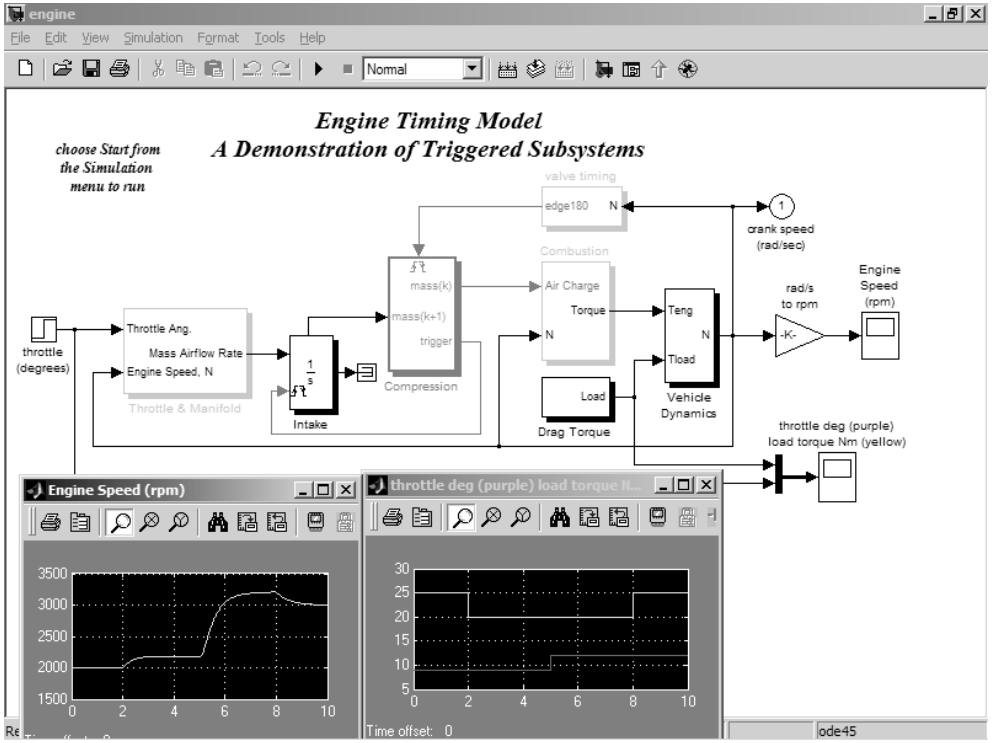


Рис. 9.46. Модель ET-системы, управляемой триггером

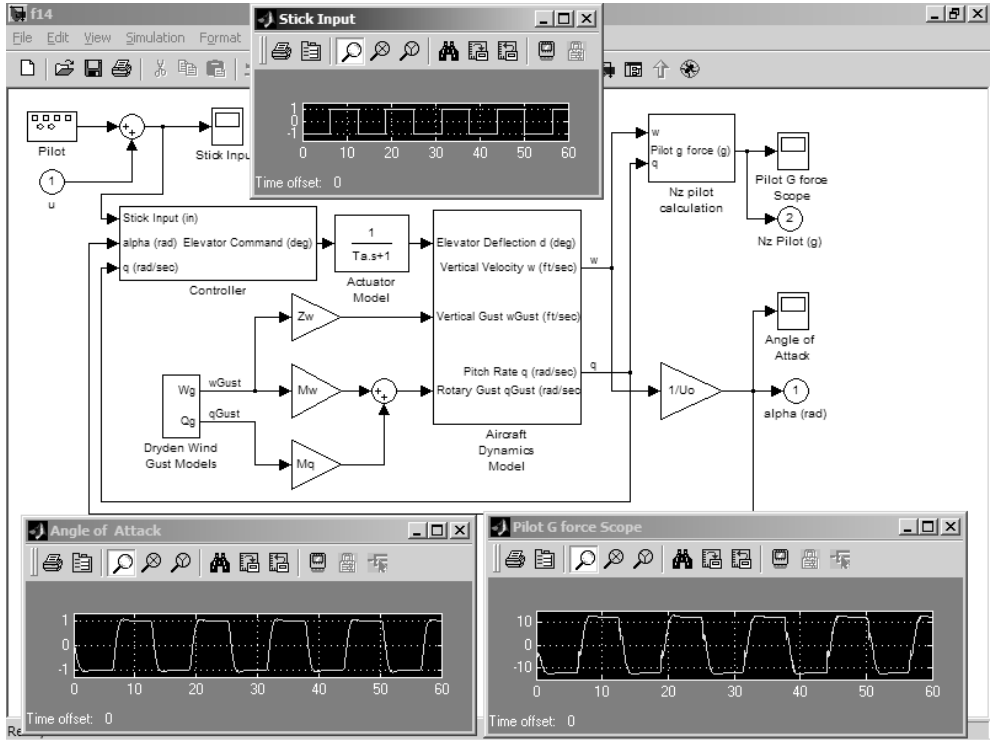


Рис. 9.47. Ранняя модель автопилота летательного аппарата F14

9.10. Моделирование ключа на мощном МДП-транзисторе

9.10.1. Построение субмодели мощного МДП-транзистора

До сих пор описывались в основном демонстрационные примеры пакета Simulink. Следует отметить, что работа с такими примерами создает совершенно ложное впечатление о простоте моделирования. Между тем построение даже простой модели требует многих часов вдумчивой работы, а самостоятельное построение моделей, подобных описанным выше, может потребовать многих дней.

Приведем пример моделирования ключа на мощном полевом МДП-транзисторе по заданной модели, которой нет в библиотеке Simulink 6.*. Эта модель адекватно описывает свойства приборов не только в ключевом, но и в линейном режиме работы. Это нужно, например, при анализе автономных энергетических устройств с линейными стабилизаторами.

За основу статической модели возьмем известные аппроксимации семейства выходных вольтамперных мощных МДП:

$$I_c = M \cdot \left(1 - e^{\frac{-kSU_{cu}}{M}}\right); \quad M = I_S \cdot \left(1 + th\left[\frac{S(U_{зu} - U_S}{I_S}\right]\right),$$

где S, I_S, U_S, k – параметры аппроксимации. Первое выражение использует экспоненциальную аппроксимацию для выходной ВАХ мощного V-МДП-транзистора, а второе выражение использует аппроксимацию гиперболическим тангенсом для передаточной характеристики.

Реализовать эти зависимости в среде Simulink возможно по-разному, и мы рассмотрим два варианта построения модели. На рис. 9.52 показана диаграмма модели MOSFET, построенной на основе блоков математических операций.

Другой вариант (рис. 9.53) реализует модель MOSFET1 с помощью компонентов **Function Block** (блока задания функциональных зависимостей). Этот вариант оказывается намного более компактным: два блока **M** и **Ic** задают соответствующие выражения для ВАХ, представленные выше.

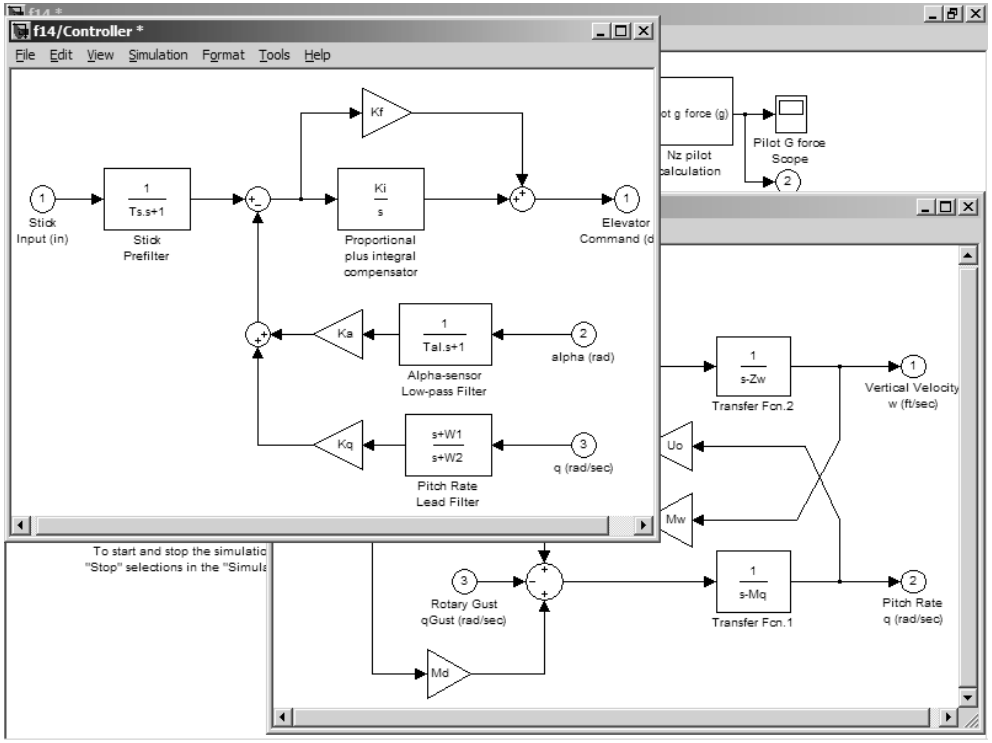


Рис. 9.48. Некоторые подсистемы модели рис. 9.47

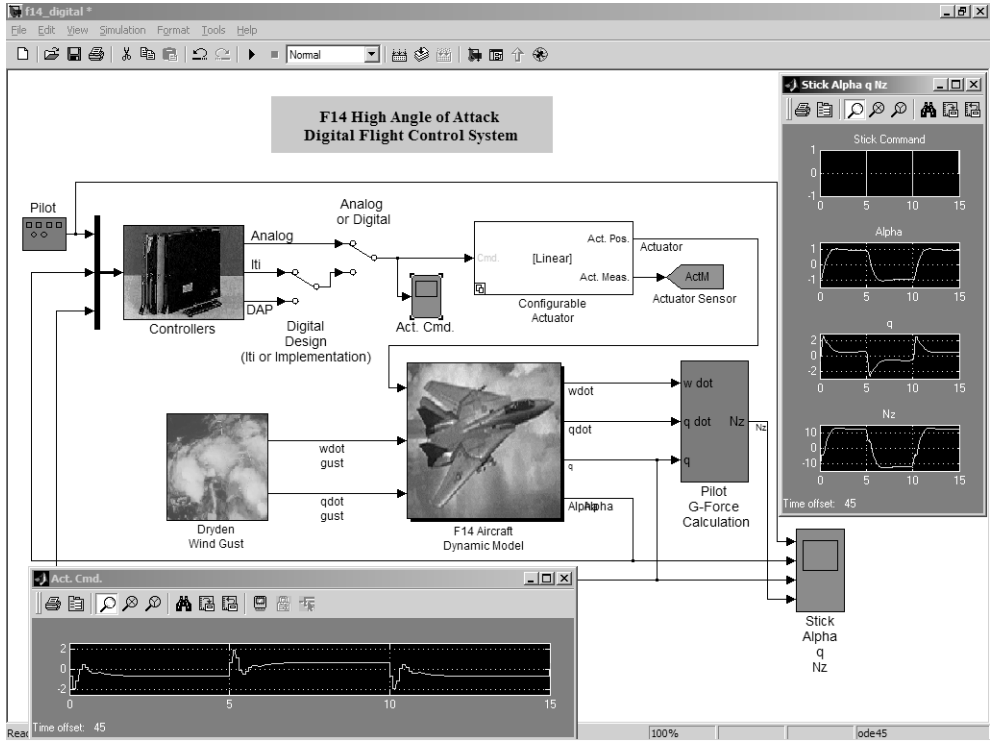


Рис. 9.49. Комбинированная модель автопилота F14 (аналоговый вариант)

9.10.2. Построение семейства ВАХ мощного МДП-транзистора

Теперь создадим виртуальный характериограф для построения семейства ВАХ мощного полевого транзистора. На рис. 9.54 показана диаграмма модели характериографа, позволяющая промоделировать снятие ВАХ импульсным методом. В этой модели блок **Signal Builder** позволяет в графическом виде задать требуемые последовательности изменения напряжений Ugs и Uds на затворе и стоке полевого транзистора.

На рис. 9.54 в левом нижнем углу представлены полученные характеристики (окно **XY Graph**). Вид импульсов, создаваемых блоком **Signal Builder**, представлен на рис. 9.55. Он хорошо иллюстрирует сущность импульсного метода снятия ВАХ.

9.10.3. Моделирование передаточной характеристики

Иногда характериографы должны строить передаточную характеристику мощного полевого транзистора – зависимость тока стока от напряжения на затворе при неизменном напряжении на стоке. Диаграмма модели, позволяющая строить такую характеристику, показана на рис. 9.56.

При построении передаточной характеристики блок создает два напряжения: линейно-нарастающее напряжение, подаваемое на затвор полевого транзистора (точнее, его модели), и постоянное напряжение, подаваемое на сток полевого транзистора. Окно с построенной передаточной характеристикой представлено на рис. 9.56 в левом нижнем углу.

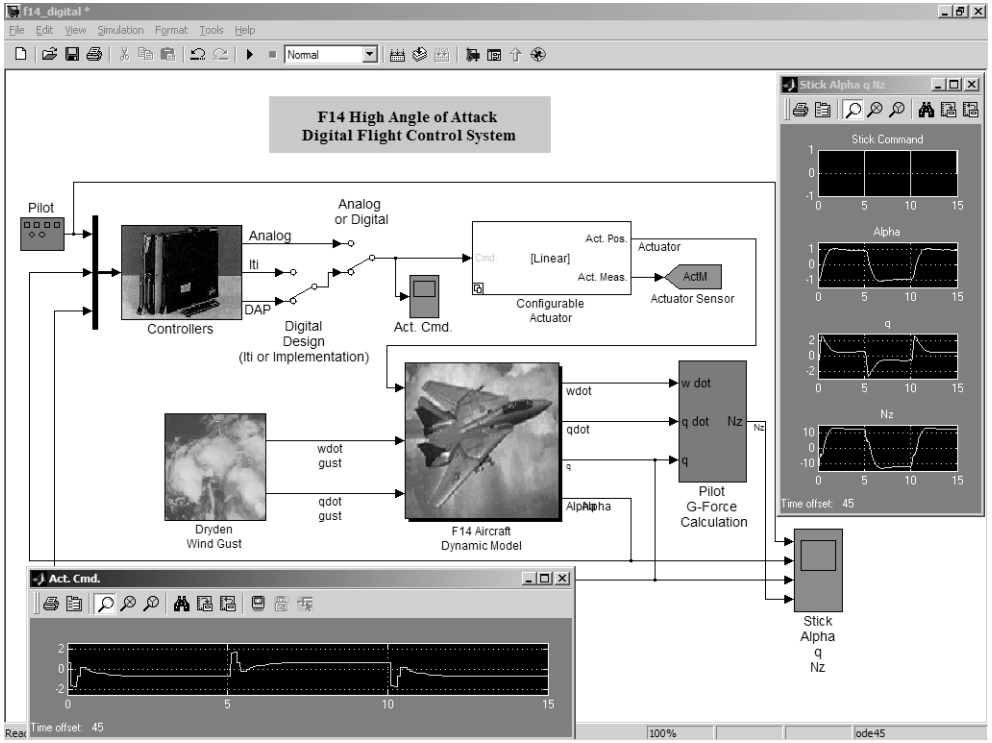


Рис. 9.50. Комбинированная модель автопилота F14 (цифровой вариант – DAP)

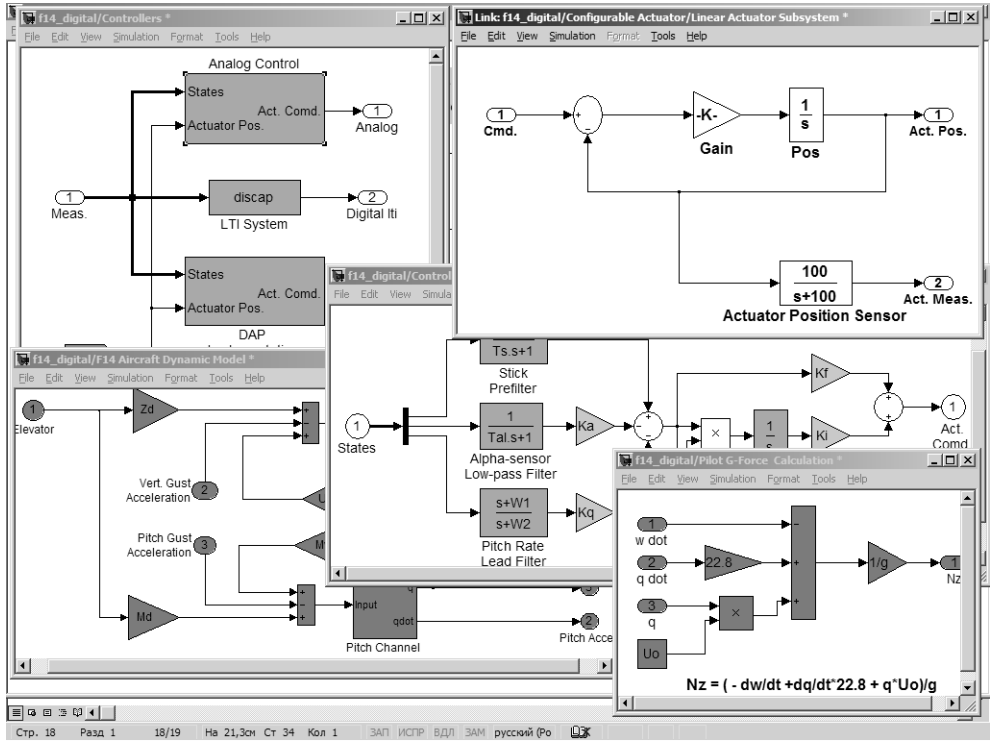


Рис. 9.51. Комбинированная модель автопилота F14 (цифровой вариант – DAP)

9.10.4. Динамическая модель мощного МДП-транзистора

Модели ключевых приборов энергетической электроники в библиотеке SimPowerSystem (см. урок 11) не учитывают динамических процессов при их переключении, у них доступно только задание параметров снабберных цепей. Это не позволяет использовать такие модели при решении серьезных задач моделирования ключевых устройств, требующих детального учета физических свойств ключевых приборов, например мощных полевых транзисторов.

Дополним полученную выше статическую модель мощного МДП-транзистора компонентами, позволяющими исследовать динамику работы прибора. В простейшем случае, ввиду очень малого времени пролета носителями области канала транзистора, достаточно учесть только паразитные межэлектродные емкости затвор–исток Cgs, затвор–сток Cgd и сток–исток Cds, приближенно считая их постоянными. На рис. 9.57 представлена модель ключа на мощном полевом транзисторе. Там же показаны блоки установки параметров для двух блоков диаграммы.

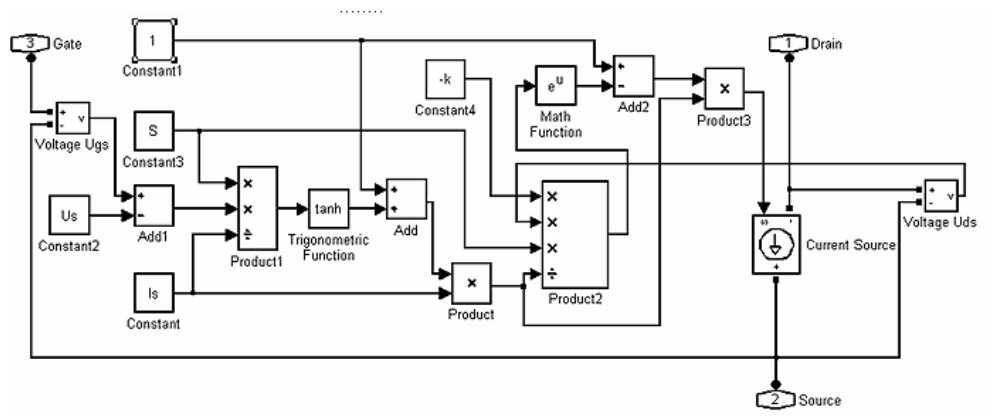


Рис. 9.52. Пример построения модели MOSFET мощного МДП-транзистора на основе математических блоков Simulink

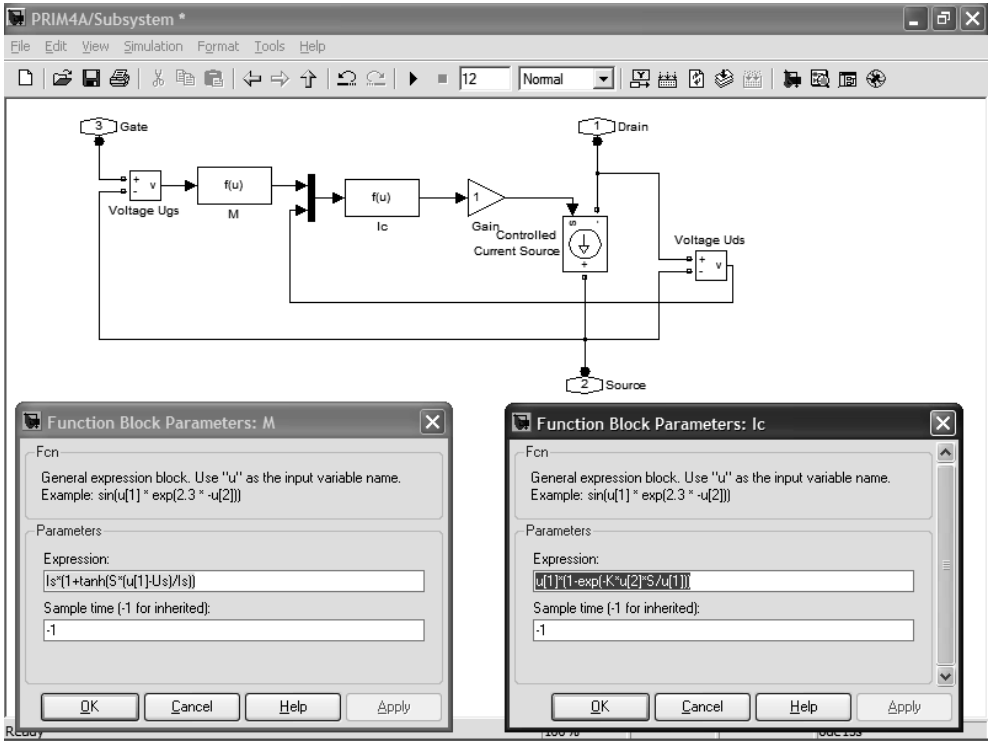


Рис. 9.53. Пример построения модели мощного МДП-транзистора MOSFET1 на основе компонентов Function Block

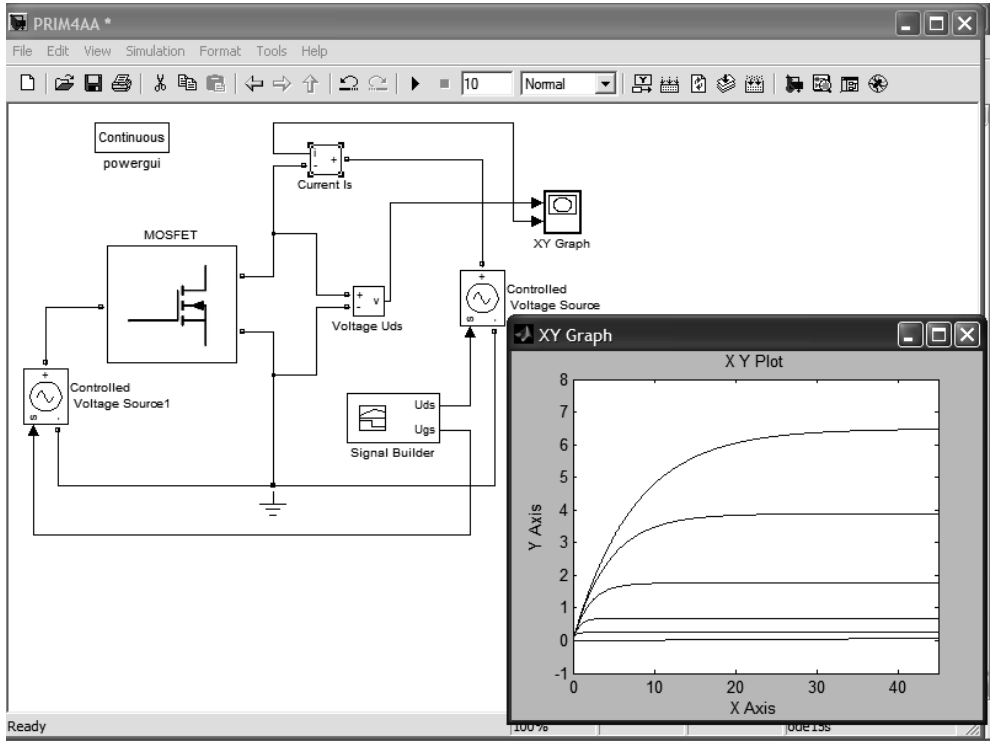


Рис. 9.54. Диаграмма улучшенной модели характериографа

Для запуска ключа обычно используются прямоугольные импульсы, подаваемые на затвор транзистора через низкоомный резистор Rg. Для создания такого импульса используется блок **Signal Boulder**, окно которого с заданным импульсом показано на рис. 9.58. В правой части этого окна показано окно осциллографа с результатами моделирования. Представлены входной импульс прямоугольной формы, импульс напряжения на затворе и импульс напряжения на коллекторе.

Емкости мощных полевых транзисторов являются нелинейными. Пренебрежение этим приводит к погрешности в десятки процентов при оценке минимальных длительностей фронтов импульсов, формируемых ключевыми схемами на мощных ПТ [70, 71]. Известно, что реальные изменения Cgs не превышают ±10...15% и можно считать Cgs=const, а для зависимостей Cgs(*U*ds) и Cds(*U*ds) у мощных МДПТ использовать аппроксимации:

$$\tilde{N}gs(Uds) = C \cdot \exp(-B \cdot Uds),$$

$$Cds(Uds) = Cds_{\min} + C \cdot \exp(-D \cdot Uds).$$

Для построения модели нелинейной управляемой емкости можно использовать субмодель с интегратором, показанную на рис. 9.59. Принцип работы модели особых пояснений не требует и основан на физическом определении емкости.

На рис. 9.60 представлена субмодель МДП-транзистора MOSFET1, более точно учитывающая его динамические свойства. Кроме нелинейности емкостей структуры прибора, блок **Transfer Fcn** позволяет дополнительно учесть инерционность процессов переключения и улучшить сходимость расчета численных методов.

9.10.5. Моделирование ключа на мощном МДП-транзисторе

На рис. 9.61 показана диаграмма модели ключа с улучшенной моделью мощного полевого транзистора с временными диаграммами работы ключа на мощном МДП-транзисторе. Они представлены в окне осциллографа.

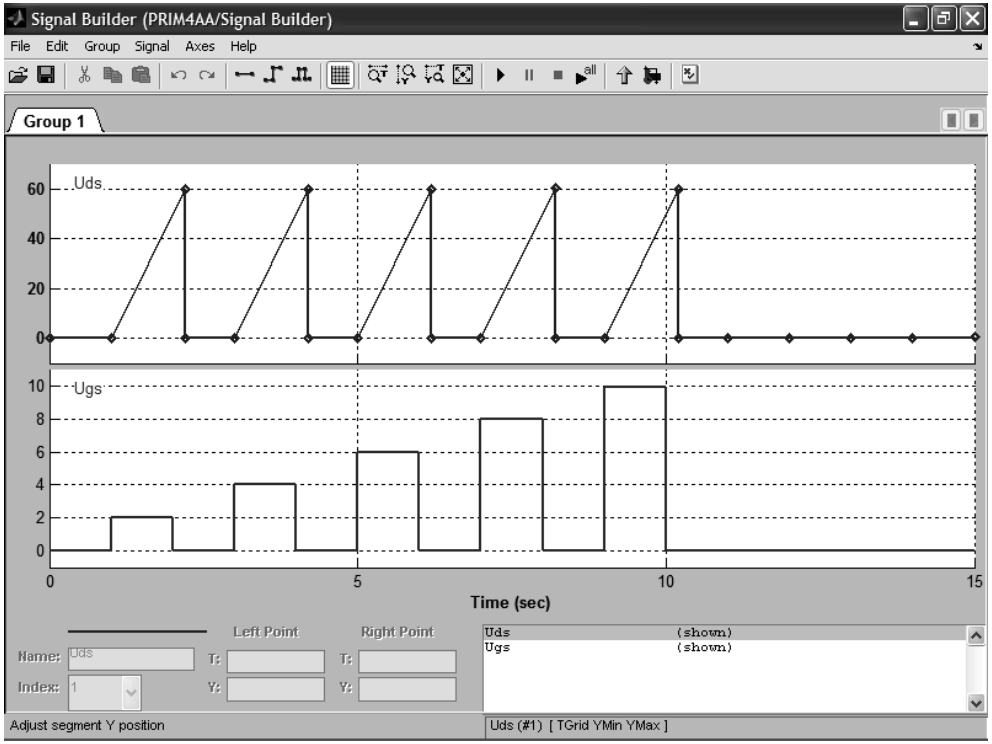


Рис. 9.55. Временные зависимости сигналов в модели характериографа, создаваемые блоком Signal Builder

Окно блока осциллографа Scope имеет ограниченные возможности по управлению форматированием графиков и экспорту изображений зависимостей в другие средства редактирования. Выйти из положения можно, выполнив следующие настройки в командном окне системы MATLAB:

```
>> set(0, "ShowHiddenHandles",'On')
>> set(gcf,'menubar','figure')
```

В результате в окне **Scope** появится стандартная строка меню графических окон типа **Figure**, дающая полный доступ к форматированию и записи графиков в различных форматах. На рис. 9.62 представлены временные диаграммы переключения ключа с использованием упрощенной и уточненной моделей МДП-транзистора в модифицированном окне блока **Scope**.

Приведенные примеры хорошо представляют возможности системы MATLAB + Simulink в решении задач моделирования нелинейных ключевых устройств по достаточно точным их моделям, учитывающим физические особенности работы приборов.

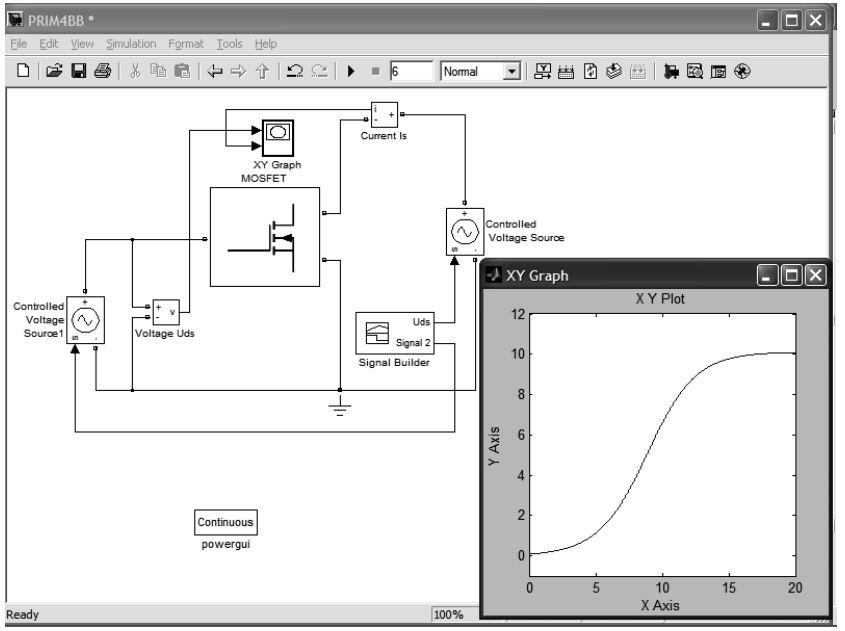


Рис. 9.56. Диаграмма модели характериографа для построения передаточной характеристики

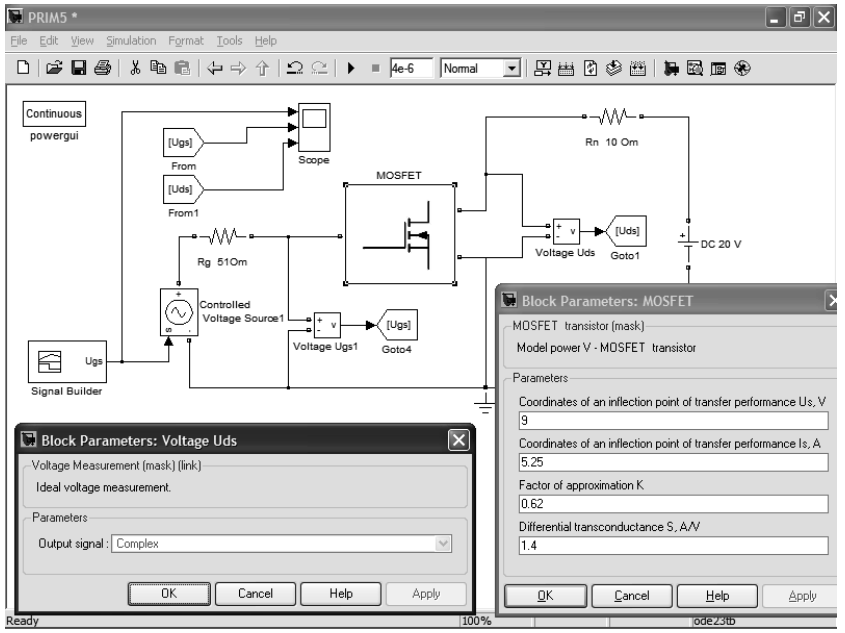


Рис. 9.57. Диаграмма модели ключа на мощном полевом транзисторе

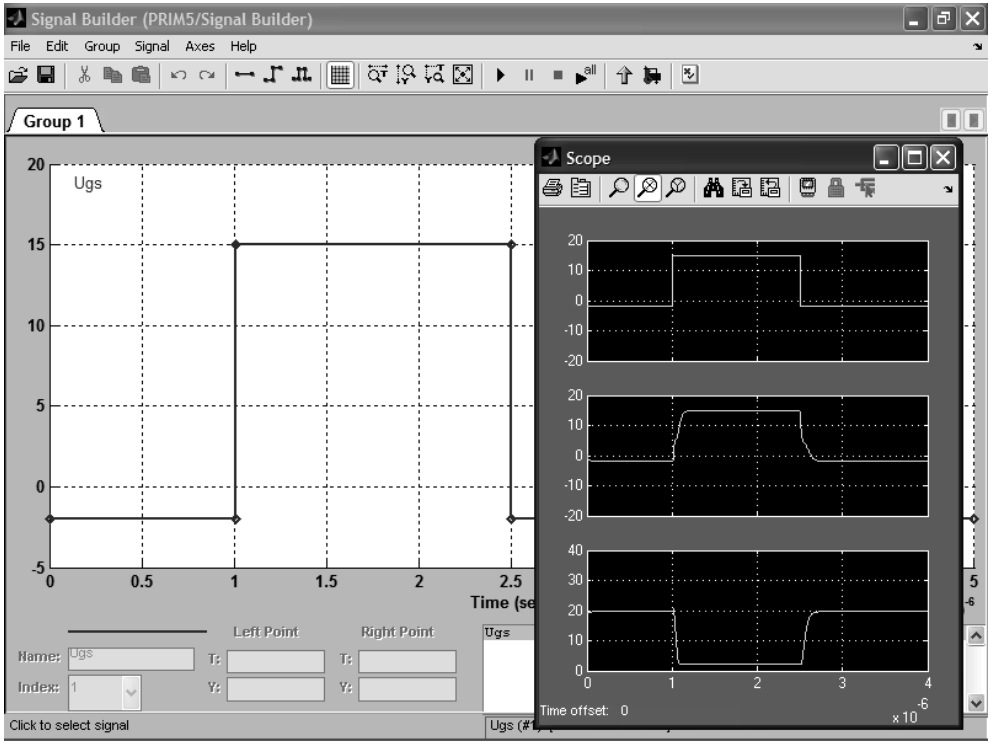


Рис. 9.58. Окна построителя сигналов и осциллографа для модели ключа на мощном полевом транзисторе

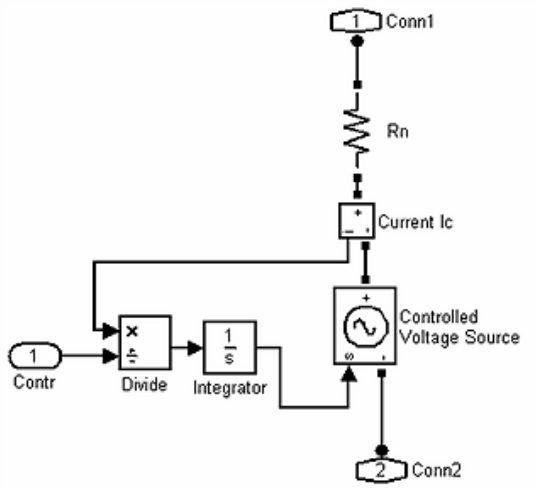


Рис. 9.59. Диаграмма субмодели нелинейной емкости

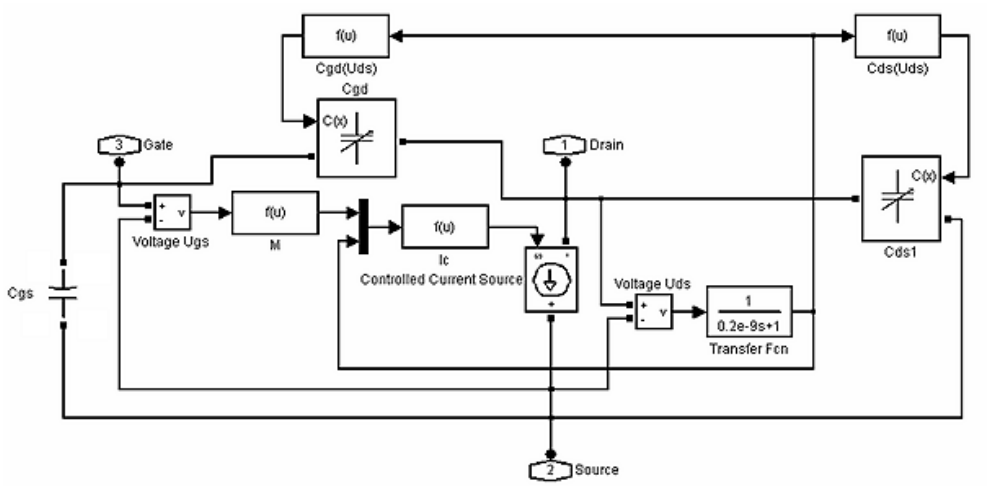


Рис. 9.60. Диаграмма уточненной субмодели MOSFET1 мощного полевого транзистора

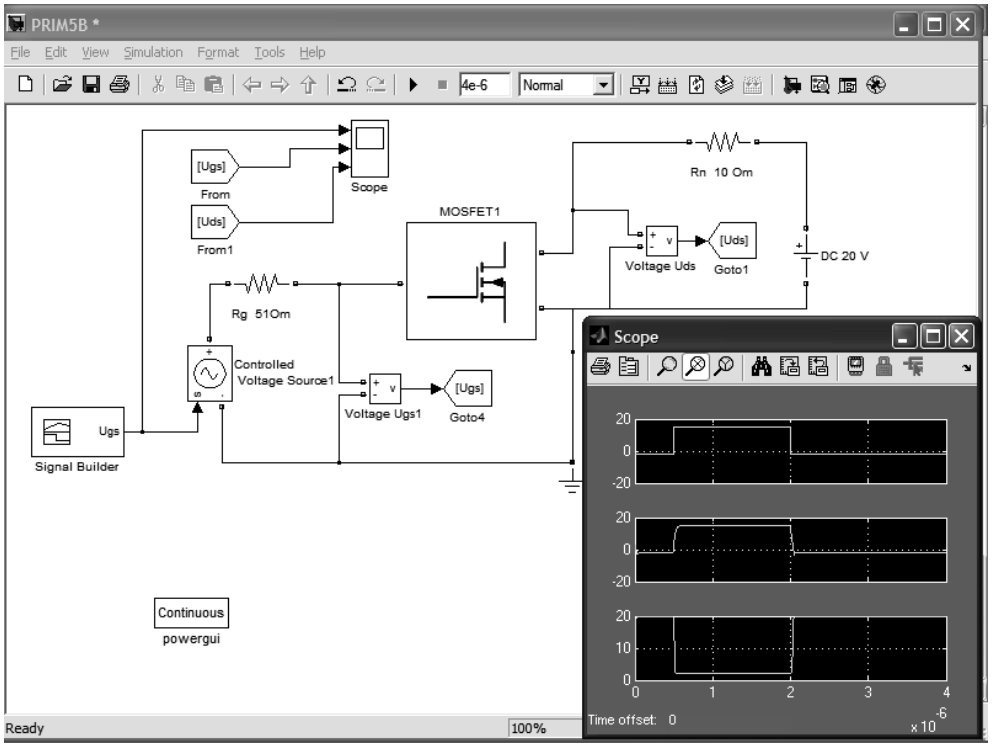


Рис. 9.61. Диаграмма модели ключа с уточненной субмоделью MOSFET1 мощного полевого транзистора и окно осциллографа с результатами моделирования

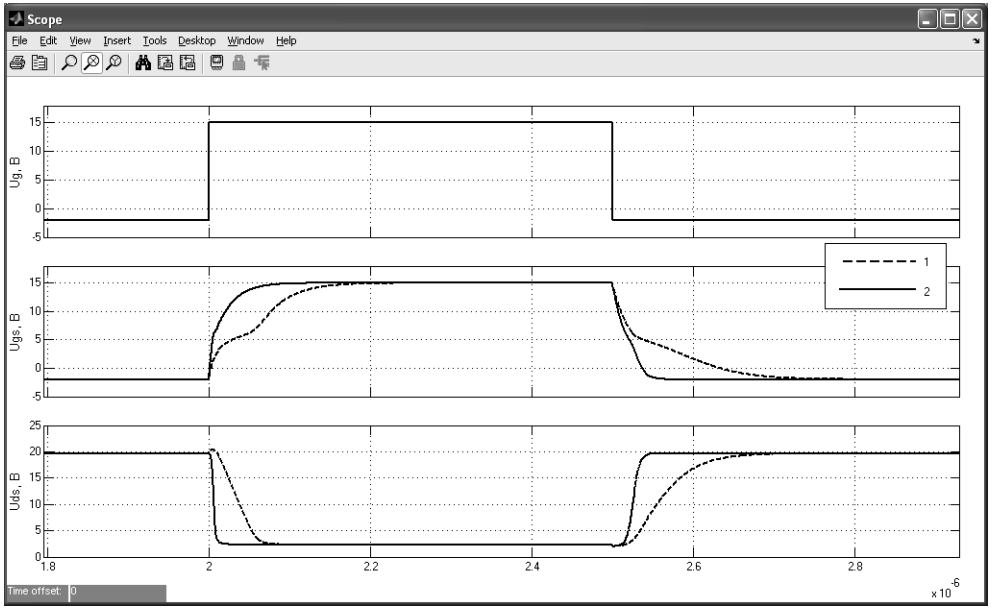


Рис. 9.62. Окно осциллографа с результатами моделирования ключей после форматирования

Урок 10

Оптимизация отклика нелинейных систем

Очень часто задачей моделирования является оптимизация тех или иных параметров систем и устройств, например отклика нелинейных систем. Возможности самого пакета Simulink в этой области ограничены. В данном уроке описаны пакеты оптимизации отклика нелинейных систем, которые входят в набор пакетов расширения Simulink, именуемый BlockSet.

10.1. Пакеты оптимизации отклика нелинейных систем

10.1.1. Назначение пакетов

Начиная с этого урока, мы рассмотрим применение с пакетом расширения Simulink некоторых наиболее важных пакетов расширения, входящих в состав инструментального ящика BlockSet.

Для динамической оптимизации систем управления в ходе их имитационного моделирования служит пакет прикладных программ построения нелинейных систем управления Nonlinear Control Design (NCD), входящий в состав инструментальных средств BlockSets Simulink 5. 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 и решение других задач управления.

Главное назначение пакета – оптимизация отклика нелинейных систем при наличии заданных ограничений на переходные процессы в таких системах.

В версиях MATLAB 6.0/6.1/6.5 используются, соответственно, версии данного пакета 5.0/6.0/7.0. Однако обновления этого пакета шли в основном в части совершенствования алгоритмов оптимизации. Поэтому излагаемый ниже материал в равной мере относится ко всем версиям этого пакета. В версии MATLAB 7 + Simulink 6 и в последующих версиях пакет Nonlinear Control Design переименован в Simulink Response Optimization. Особенности этого пакета расширения описаны в конце данного урока.

10.1.2. Состав блоков пакетов

Набор блоков пакетов оптимизации моделируемых нелинейных систем содержит всего три блока: **CRMS** (Continue RMS), **DRMS** (Discrete RMS) и **NCD Output** (в Simulink Response Optimization блок **Signal Constraint**). Блок **CRMS** реализует математическую зависимость

$$y(t)=\sqrt{\frac{1}{t}\int_0^t u^2(\tau)d\tau}\ \text{ при }\ t>0,$$

где $u(t)$ – входной сигнал блока, $y(t)$ – его выходной сигнал. Для стационарных случайных процессов с нулевым математическим ожиданием выходной сигнал блока при $t\to\infty$ является среднеквадратическим отклонением.

Блок **DRMS**, по сути, реализует ту же зависимость, что и блок **CRMS**, но для сигналов, определенных в дискретные моменты времени:

$$y(n)=\sqrt{\frac{1}{n}\sum_{k=0}^{n}u^2(k)}\ \text{ при } n>0.$$

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.

10.1.3. 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. 10.1). С помощью блока **Mux** их выходы подключаются к входу виртуального осциллографа.

Активизируем блок осциллографа **Scope** (двойным щелчком мыши) и запустим процесс моделирования. Его результат отображен на осциллограммах рис. 10.1, 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.

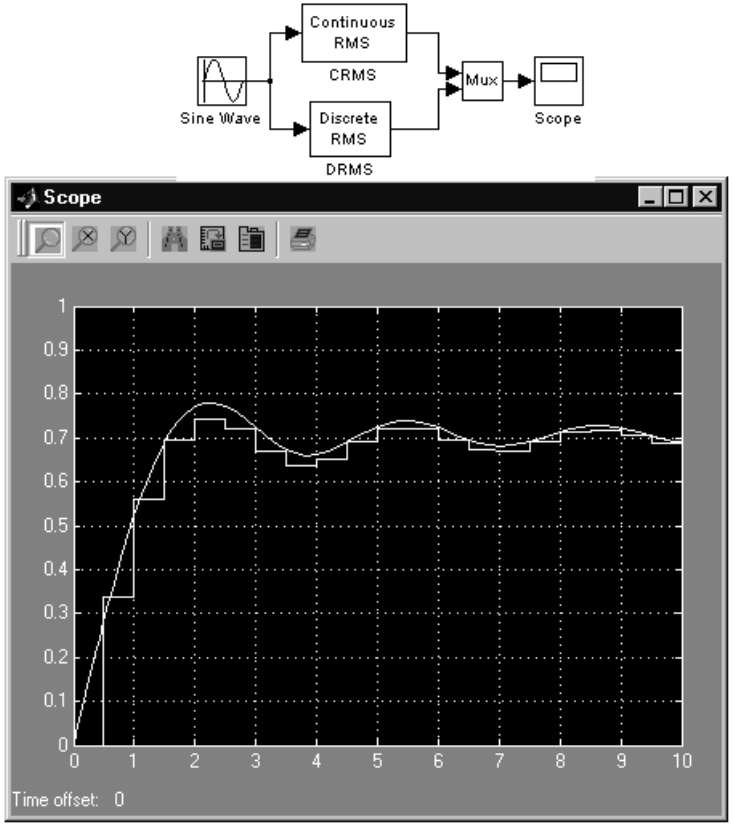


*Рис. 10.1. Модель, иллюстрирующая функционирование блоков **CRMS** и **DRMS***

10.2. Оптимизация нелинейных систем с помощью пакета NCD

10.2.1. Оптимизация коэффициента передачи И-регулятора

Команда `ncdtut1` для пакета Nonlinear Control Design открывает окно Simulink с моделью замкнутой системы автоматического регулирования – рис. 10.2.

Модель состоит из следующих компонентов:

- объекта регулирования с передаточной функцией

$$\frac{\omega_0^2 e^{-p\tau}}{p^2 + 2\omega_0\xi p + \omega_0^2},$$

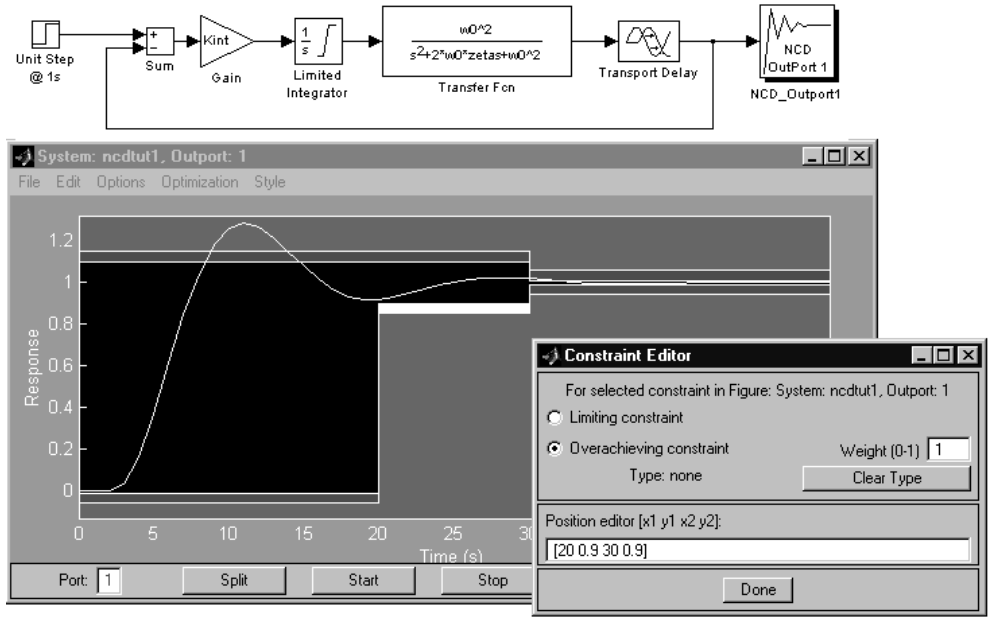


Рис. 10.2. Модель системы И-регулятора

который в модели отображен звеном второго порядка (блок **Transfer Fcn**) и звеном задержки (блок **Transport Delay**) со значениями $\tau = 1$ с, $\omega_0 = 1$ с$^{-1}$, $\xi = 1$ и обозначениями **w0** = ω_0, **zetas** = ξ;

- интегрального И-регулятора, в состав которого входят последовательно соединенные пропорциональное звено с коэффициентом передачи **Kint** и интегрирующее звено с ограничением выходного сигнала (блок **Limited Integrator**);
- контура обратной связи и звена 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 20 с;
- длительность переходного процесса – не более 30 с.

При решении задачи оптимизации учтем, что первые два ее этапа (см. выше описание блока **NCD Output**) 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 и с помощью мыши передвинем границы, задающие ограничения, в нужное положение. 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. 10.3).

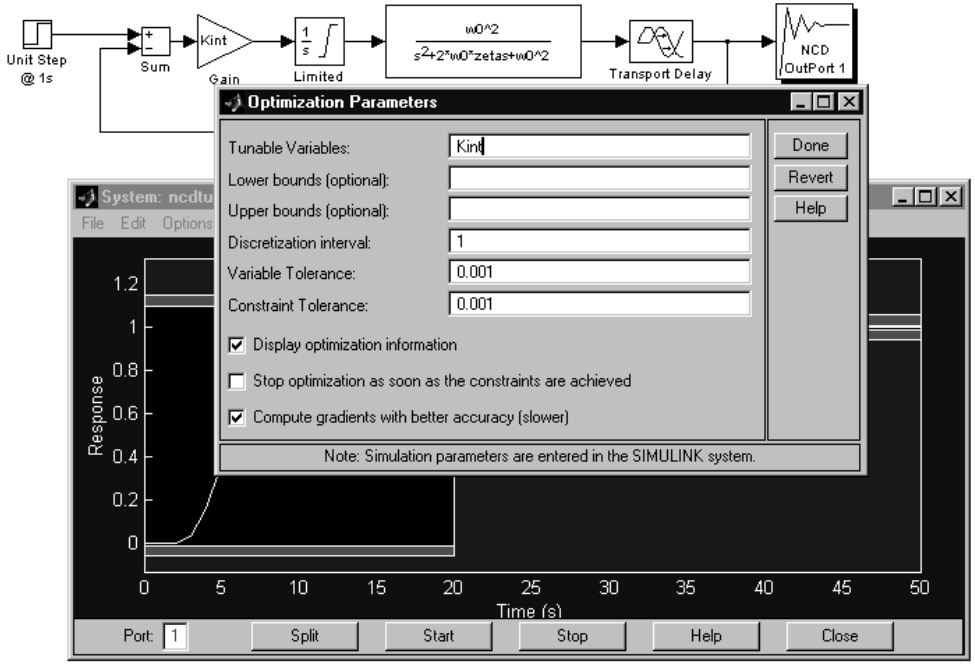


Рис. 10.3. Настройка параметров оптимизации интервала дискретизации

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. 10.3).

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**.

Теперь все готово для решения задачи. Процесс поиска решения можно запустить нажатием кнопки **Start** в панели инструментов окна модели Simulink. Результат поиска иллюстрируется начальной и конечной формами переходного

процесса, представленными на рис. 10.4. В окне MATLAB также выводятся данные о ходе оптимизации.

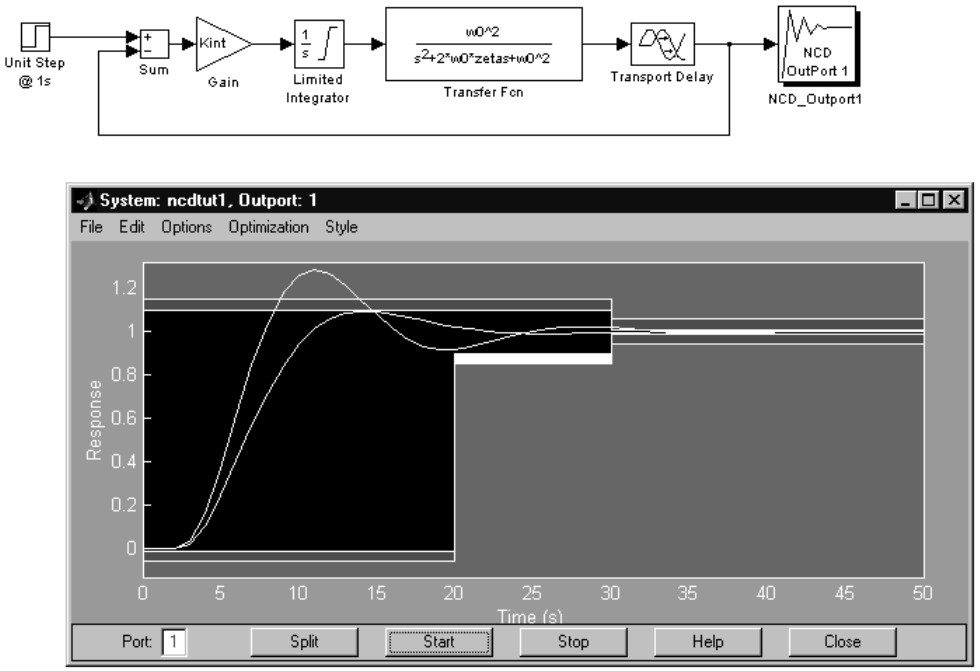


Рис. 10.4. Иллюстрация процесса оптимизации

По рис. 10.4 видно, что оптимизированная переходная характеристика полностью соответствует заданным ограничениям. Найдем оптимальную величину параметра **Kint**:

```
>> Kint
Kint =  0.1961
```

Задачу оптимизации можно усложнить, введя неопределенные параметры, точные значения которых неизвестны или могут претерпевать изменения. 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. 10.5.

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**

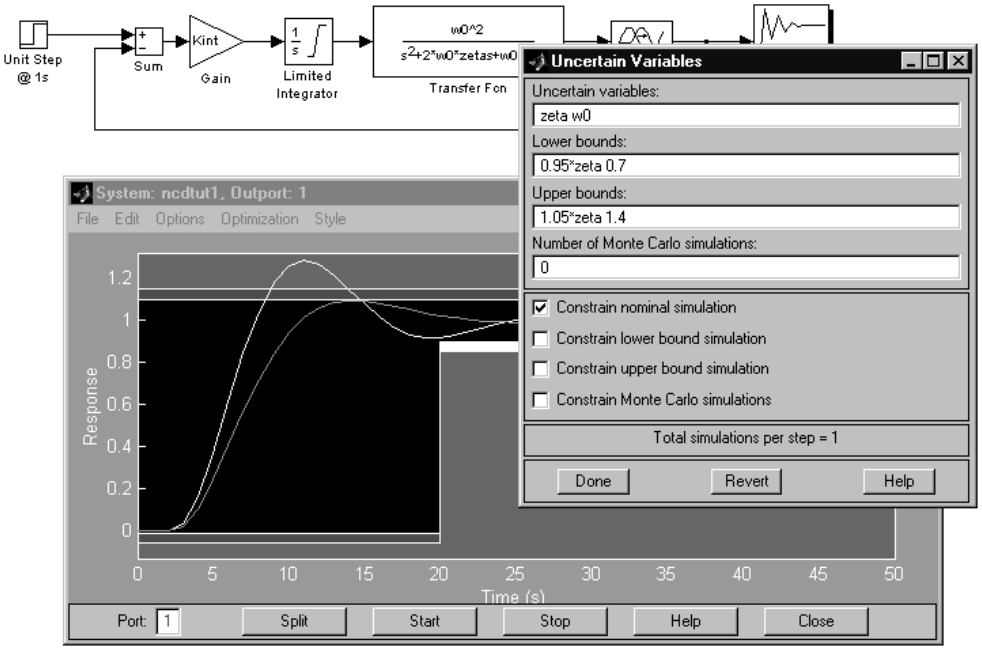


Рис. 10.5. Пример задания неопределенных переменных

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. Нажмем для этого кнопку **Start** в панели инструментов окна модели Simulink, и результат выполнения оптимизации появится на временной диаграмме (рис. 10.6).

На данном рисунке одна группа кривых соответствует переходным процессам в системе в том случае, когда неопределенные параметры принимают нижнее и верхнее граничные значения, а коэффициент **Kint** – начальное значение; другая группа кривых соответствует 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**).

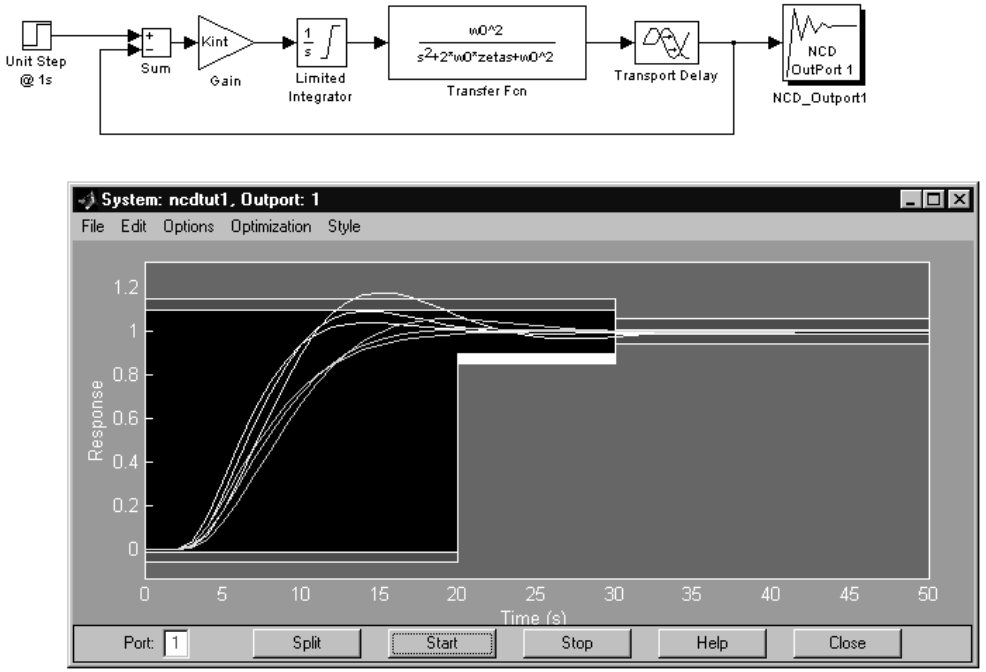


Рис. 10.6. Иллюстрация процесса оптимизации при наличии неопределенных параметров

10.2.2. 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. 6.4), в котором можно указать параметры входного сигнала системы для вывода соответствующего графика. Никакого влияния на процессы моделирования и оптимизации данные параметры не оказывают;

- **Step response (Характеристики переходного процесса)**. Выбор этого пункта приведет к открытию диалогового окна **Step response** (рис. 10.7), 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**);
- **Time range (Временной диапазон)**. Выбор данного пункта приводит к простому открытию диалогового окна **Time Axis Limits**, в котором можно задать или изменить диапазон времени моделирования и метку оси времени, то есть параметры оси абсцисс;
- **Y-Axis (Ось Y)**. То же для оси ординат;
- **Refresh (Обновить)** – перерисовать все временные ограничения.

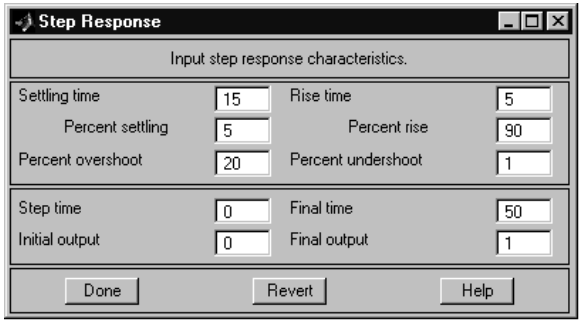


Рис. 10.7. Окно задания характеристик переходного процесса

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 10,5 градуса;
- **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.

10.2.3. 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. 10.8).

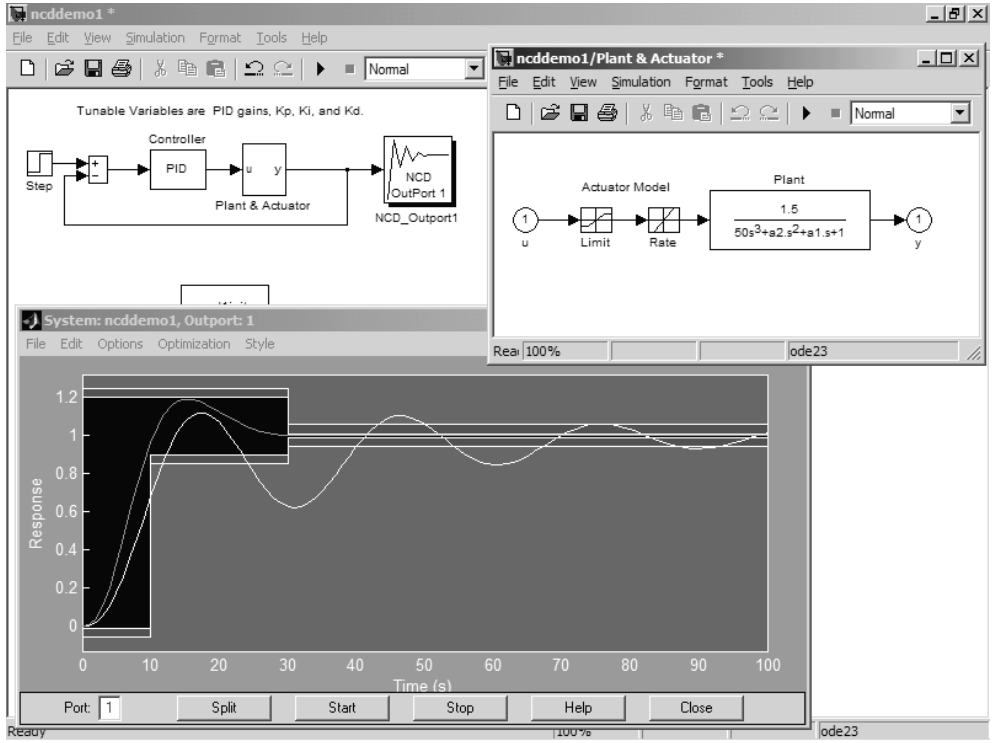


Рис. 10.8. Моделирование PID-регулятора

Основными элементами замкнутой системы PID-регулятора являются:

- объект регулирования (блок **Plant & Actuator**);
- PID-регулятор (**Controller**) на основе дифференцирующего устройства и интегратора;
- 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 & Ac-**

tuator двойным щелчком мыши. В состав объекта регулирования (см. его окно на рис. 10.8 справа) 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

$$W(p) = \frac{1.5}{50p^3 + a_2 \cdot p^2 + a_1 \cdot p + 1},$$

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** = 1,96810. Такие значения выбраны в соответствии с методикой настройки ПИД-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**/(5Ч**Pu**), **Kd** = 3·**Ku**·**Pu**/40.

Запустим процесс оптимизации, нажав кнопку **Start**. Полученный результат отражен на рис. 10.10. Найденные оптимальные значения (в командном режиме работы MATLAB) при этом равны:

```
>> Kp
Kp = 1.3366
>> Ki
Ki =  0.1548
>> Kd
Kd =  10.3318
```

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. 10.9.

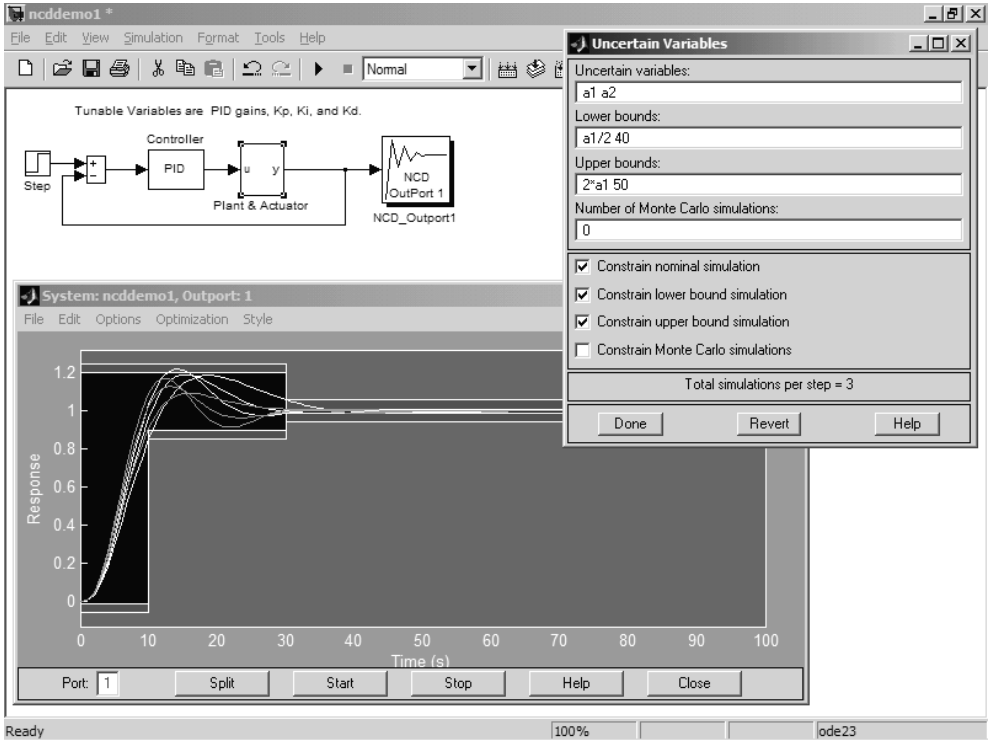


Рис. 10.9. Иллюстрация процесса оптимизации при неопределенных параметрах a1 и a2

В командном режиме вычислений находим:

```
>> Kp
Kp =  1.6740
>> Ki
Ki =   0.1273
>> Kd
Kd =  10.2597
```

По сравнению с предыдущим случаем здесь имеются определенные отличия. Процесс исследования системы можно продолжить, вводя, например, дополнительные ограничения или изменяя введенные и т. п.

10.2.4. 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. 10.10).

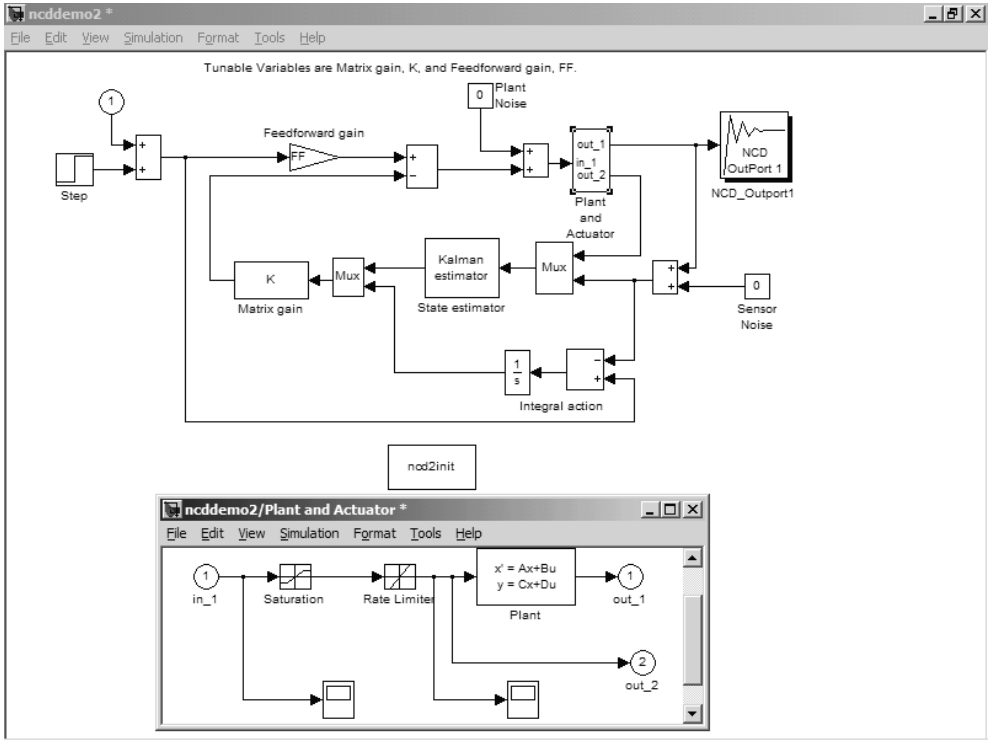


Рис. 10.10. Модель системы регулирования

Объект регулирования (**Plant and Actuator**) представляет собой (см. рис. 10.10 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:

$$\dot{x} = \begin{bmatrix} -1.0285 & 0.9853 & -0.9413 & 0.0927 \\ -1.2903 & -1.0957 & 2.8689 & 4.7950 \\ 0.1871 & -3.8184 & -2.0788 & -0.9781 \\ 0.4069 & -4.1636 & 2.5407 & -1.4236 \end{bmatrix} x + \begin{bmatrix} 0 \\ 6.6389 \\ 0 \\ 0 \end{bmatrix} u = Ax + Bu,$$

$$y = [-1.7786 \quad 1.1390 \quad 0 \quad -1.0294]x = Cx.$$

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 2 с;
- длительность переходного процесса – не более 4 с.

Результат моделирования представлен на рис. 10.11.

При исходных значениях параметров системы вид ее переходного процесса был весьма далек от желаемого, но итоговый переходный процесс укладывается

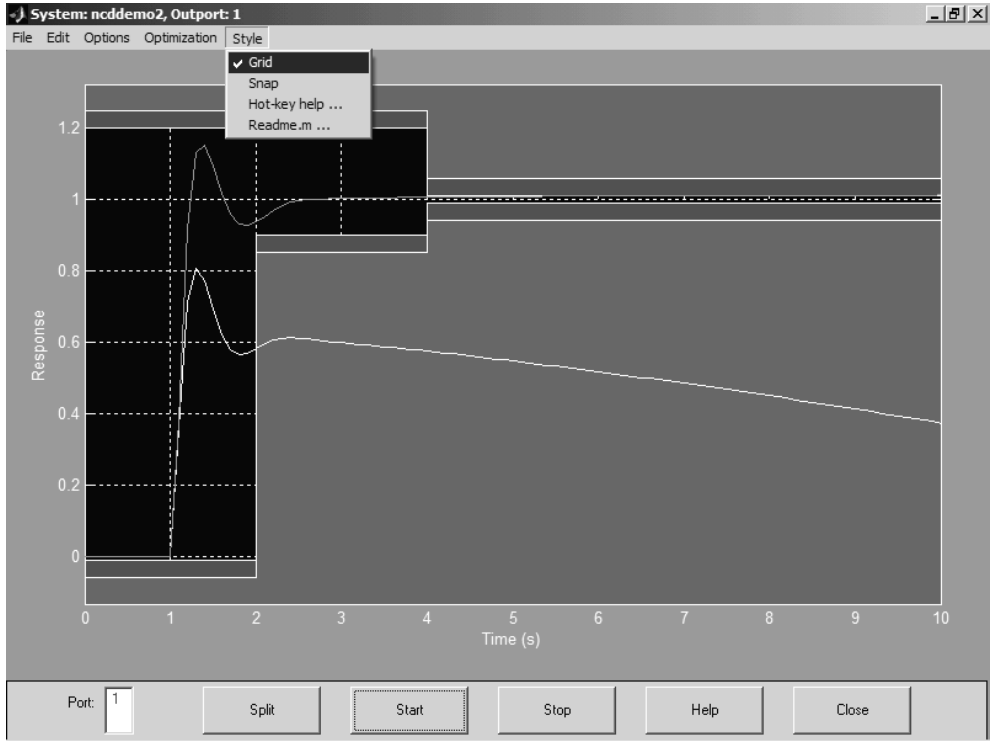


Рис. 10.11. Иллюстрация процесса оптимизации

в заданные временные ограничения. Найденные оптимальные значения параметров можно получить в режиме командной строки MATLAB:

```
>> K
K=
   -1.0585    0.9378    0.0188   -0.2413    0.0786
>> FF
FF = 1.2729
```

Заметим, что при установленных параметрах (см. рис. 10.11) найденное решение соответствует номинальным значениям элементов матрицы *A*. Для исследования поведения системы с учетом неопределенности данной матрицы необходимо выполнить действия, описанные в предыдущем примере.

10.2.5. Настройка параметров ПИ-регулятора для многомерного объекта

Наконец, еще одним примером (отраженным в файле **ncddemo3**) является пример настройки параметров пропорционально-интегрального регулятора (ПИ-регулятора) для многомерного объекта. В качестве такого объекта 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. 10.12.

В данном случае параметрами оптимизации являются матрицы коэффициентов **Ki** и **Kp** двухканального ПИ-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, отображаемые рис. 10.12, следует признать очень качественными (здесь оптимизация проведена для номинальных значений элементов матрицы *A*).

Найденные оптимальные значения таковы:

```
>> Ki
Ki =
   12.5209   -2.5168
   46.2438  -51.7909
>> Kp
Kp =
    1.6317    6.0054
  127.0848  -510.4045
```

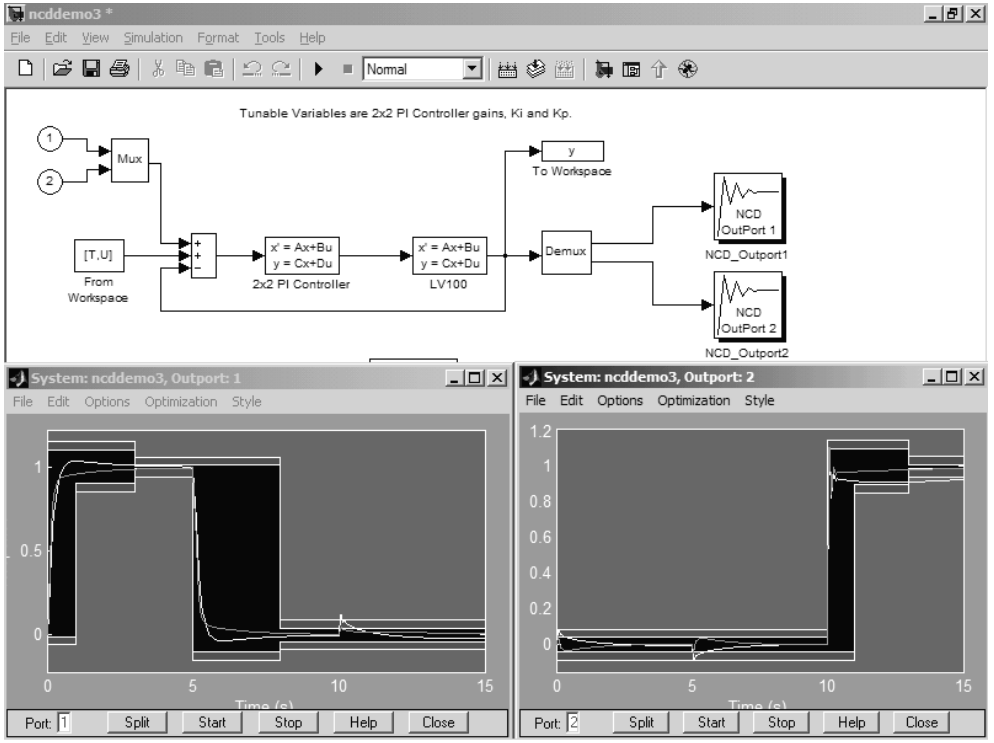



Рис. 10.12. Системы с многомерным объектом регулирования

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`.

10.2.6. Особенности решаемых оптимизационных 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.

10.2.7. Функции и команды NCD Blockset

В рассматриваемый пакет входит следующий набор функций и команд, выполняемых в режиме командной строки MATLAB.

Функции управления диалоговыми окнами:

- `coneddlg` – вызов окна редактора ограничений;
- `paramdlg` – вызов окна задания настраиваемых параметров;
- `rangedlg` – вызов окна задания диапазонов осей координат;
- `refdlg` – вызов окна задания параметров опорного сигнала;
- `stepdlg` – вызов окна задания параметров переходного процесса системы;
- `uncerdlg` – вызов окна задания неопределенных переменных.

Функции данной группы, выполняемые при значении аргумента, равном нулю (например, `stepdlg(0)`), дублируют соответствующие команды меню окна блока **NCD Output**. Функции выполняются только при открытом данном окне.

Функции главного интерфейса **NCD Blockset**:

- `ncdblock` – вызов блоков **NCD**;
- `slblocks` – определение библиотеки блоков **NCD Blockset**;
- `optblock` – открытие блока **NCD Output**;
- `optfig` – создание временных ограничений.

Например, выполнение функции `ncdblock` приведет к появлению окна Simulink с блоками **NCD**.

Функции оптимизации:

- `initresp` – возвращает график временной характеристики при начальных значениях параметров модели;
- `costfun` – вычисляет функцию потерь;
- `gradfun` – вычисляет градиент функции потерь;
- `nlinopt` – запускает алгоритм оптимизации;
- `copymdl` – создает дополнительную модель, используемую для нахождения градиента;
- `tvarset` – S-функция для задания настраиваемых параметров.

Первая из указанных функций дублирует соответствующую команду меню главного окна блока **NCD Output**, остальные используются в процессе оптимизации и, по сути, являются вызываемыми подпрограммами данного блока.

Функции вызова демонстрационных примеров:

- `ncddemo` – библиотека демонстрационных примеров;
- `ncddemo1` – оптимизация ПИД-регулятора;
- `ncddemo2` – оптимизация регулятора сложной структуры;
- `ncddemo3` – оптимизация ПИ-регулятора для многомерной системы;
- `ncddemo4` – оптимизация системы управления перевернутым маятником;
- `rmsdemo` – демонстрация блоков **CRMS** и **DRMS**.

Функции обучения:

- `ncdtut1` – пример проектирования системы управления;
- `ncdtut2` – пример идентификации системы;
- `ncdtut2old` – вариант примера идентификации системы.

Утилиты для функций демонстрации и обучения:

- `ncd1init` – задание ограничений и начальных условий для `ncddemo1`;
- `ncd2init` – задание ограничений и начальных условий для `ncddemo2`;
- `ncd3init` – задание ограничений и начальных условий для `ncddemo3`;
- `ncd4init` – задание ограничений и начальных условий для `ncddemo4`;
- `penddata` – задание ограничений и начальных условий для `ncdtut2`.

Файлы описаний (текстовые файлы с расширением **.hlp**):

- `hotkey` – информация об используемых комбинациях клавиш;
- `mainncd` – общее описание блоков **NCD**;
- `paramdlg` – описание окна задания настраиваемых параметров;
- `readncd` – то же, что файл **Readme.m**;
- `stepdlg` – описание диалогового окна для задания параметров переходного процесса системы;
- `uncerdlg` – описание диалогового окна задания неопределенных переменных.

В пакет входит также набор утилит оптимизации и графического интерфейса пользователя (GUI), используемых в качестве подпрограмм основной программы блока **NCD Output**.

Более подробную информацию о каждой из перечисленных функций можно получить, вызвав в режиме командной строки MATLAB команду `help имя_функции`.

10.3. Новый пакет расширения Simulink Response Optimization

10.3.1. Назначение пакета расширения Simulink Response Optimization

Новый пакет расширения Simulink Response Optimization 2.* появился в системе MATLAB 7.0 (R14) + Simulink 6.0 (и в последующих версиях) и заменил подобный пакет Nonlinear Control Design, описанный выше. Поскольку пакеты по основным возможностям идентичны, ограничимся кратким описанием отличий пакета Simulink Response Optimization и его новых примеров. В версиях MATLAB 2006a/2006b/2007a используются обновленные версии пакета Simulink Response Optimization 3.0/3.1/3.1.1. В разделе справки Realise Note по новейшей версии 3.1.1 этого пакета есть подробное описание деталей каждой версии.

По пакету имеется обширная справка, окно которой представлено на рис. 10.13. Здесь справа представлен раздел по функциям пакета, разделенным на категории.

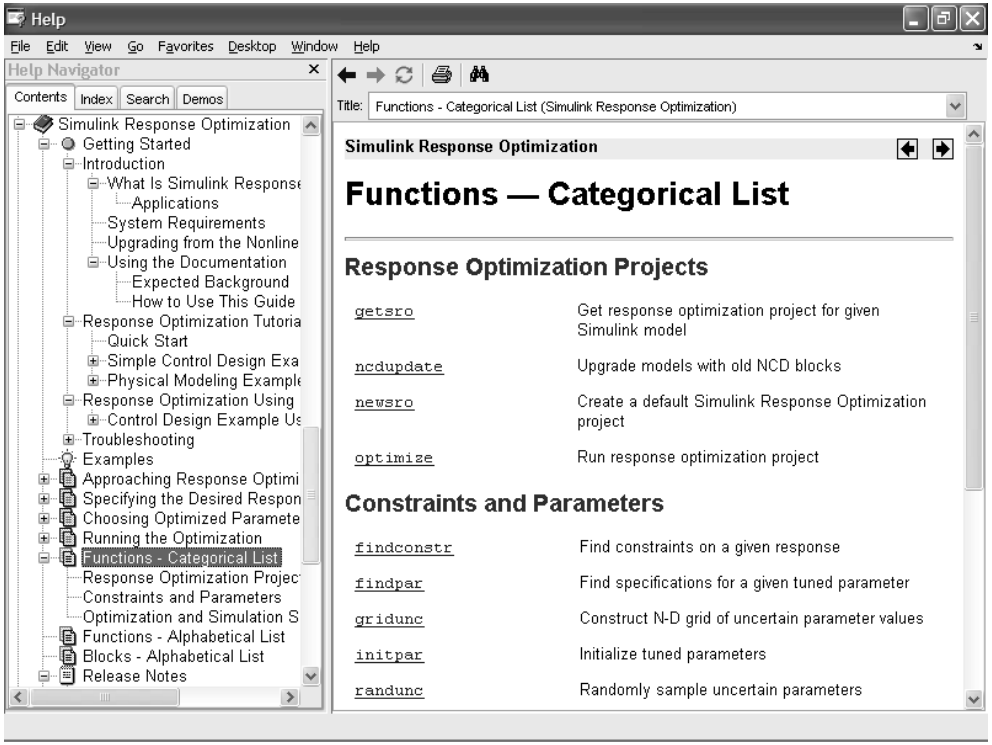


Рис. 10.13. Окно справки по пакету расширения Simulink Response Optimization

Функции предназначены для создания MATLAB-программ для управления процессом анализа и оптимизации нелинейных систем. Их можно использовать в программах пользователя, но, как правило, нужные программы составляются автоматически в ходе визуально-ориентированной подготовки диаграммы моделей. Описание функций было приведено ниже.

Как и у других пакетов расширения, доступ к демонстрационным примерам осуществлятся с вкладки **Demos** справки – рис. 10.14.

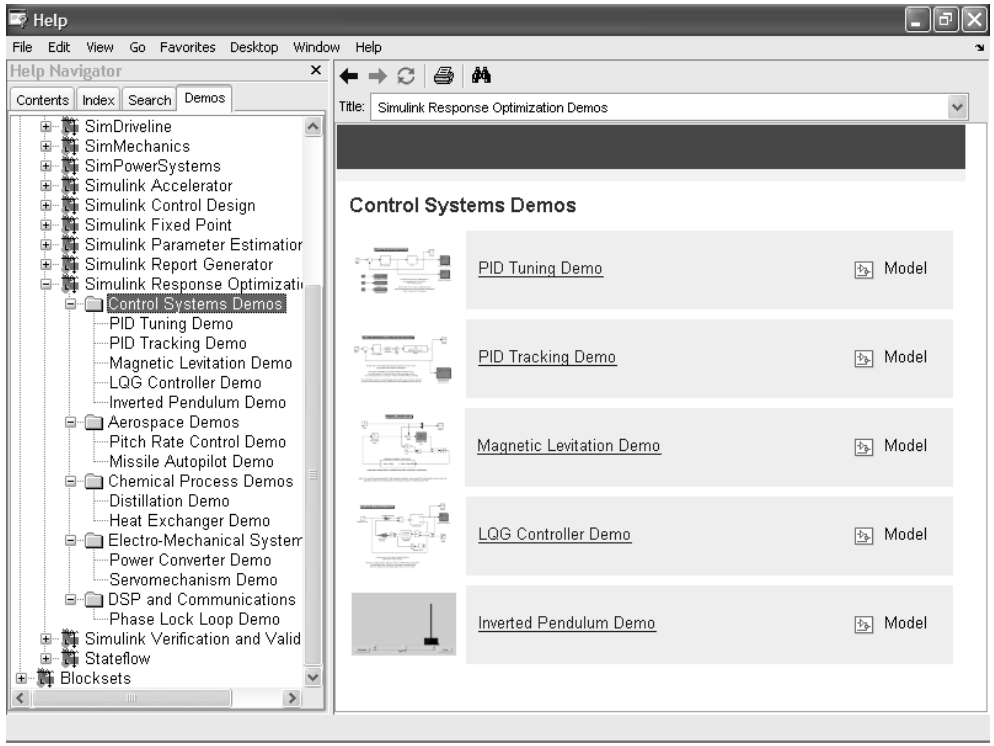


Рис. 10.14. Окно демонстрационных примеров пакета Simulink Response Optimization

Ряд примеров практически аналогичен тем, которые давались при описании пакета Nonlinear Control Design. Например, команда

```
>> rmsdemo
```

вызовет появление окна, абсолютно идентичного показанному на рис. 10.1 для пакета Nonlinear Control Design. Однако в целом набор примеров несколько расширен и дает хорошее представление о возможностях пакета. Ниже мы рассмотрим лишь несколько характерных примеров, представляющих возможности пакета расширения Simulink Response Optimization в части оптимизации отклика нелинейных систем.

10.3.2. Оптимизация системы с PID-контроллером

Рассмотрим уже знакомый нам пример – оптимизации системы с PID-регулятором. В списке демонстрационных примеров этот пример называется PID Tracking Demo. Активизировав его гиперссылку, можно вывести окно диаграммы, представленное на рис. 10.15. В нем дано также окно с диаграммой субблока (PID-контроллера) и осциллограммой переходного процесса после оптимизации.

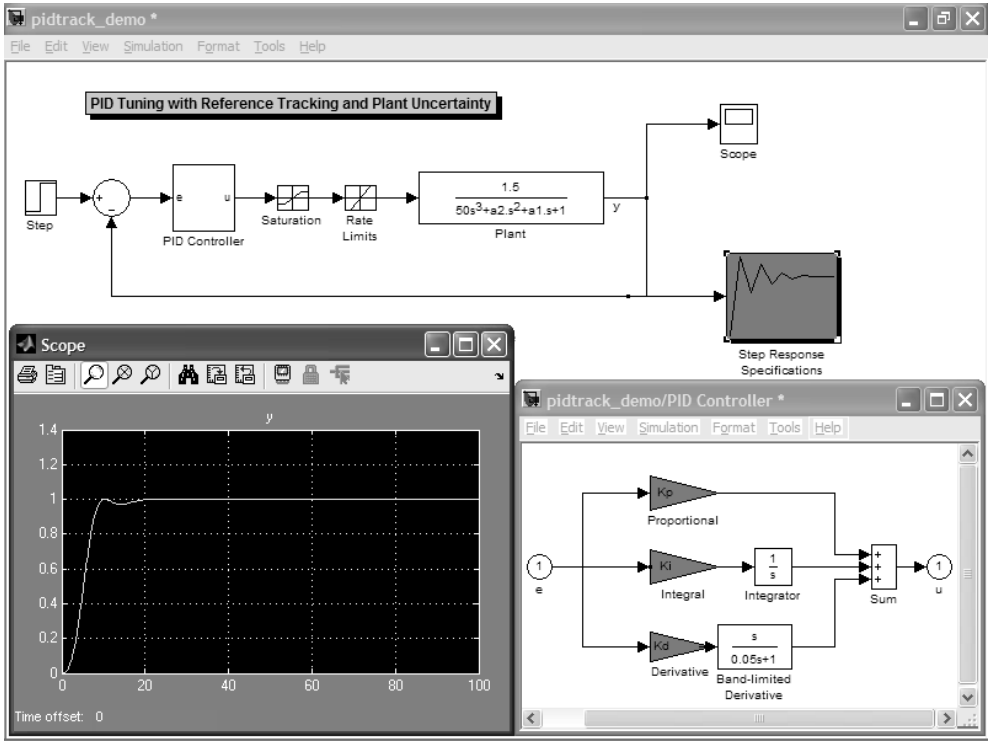


Рис. 10.15. Окно диаграммы примера PID Tracking Demo

Этот пример выполняет оптимизацию в итерационном режиме и выводит все данные, включая промежуточные (есть также пример PID Tuning Demo, в котором вывод промежуточных результатов оптимизации не предусмотрен, и пример srotut1 с запуском из командной строки Simulink). При пуске моделирования выводится окно **Optimization Progress** (рис. 10.16), в котором можно наблюдать за ходом оптимизации и контролировать параметры модели.

После оптимизации можно наблюдать окно, показанное на рис. 10.17. В нем представлены как окончательная кривая переходного процесса (она же показывается в окне контрольного осциллографа), так и кривые переходных процессов на всех итерациях. Естественно, показаны и задаваемые нормы для переходных процессов.

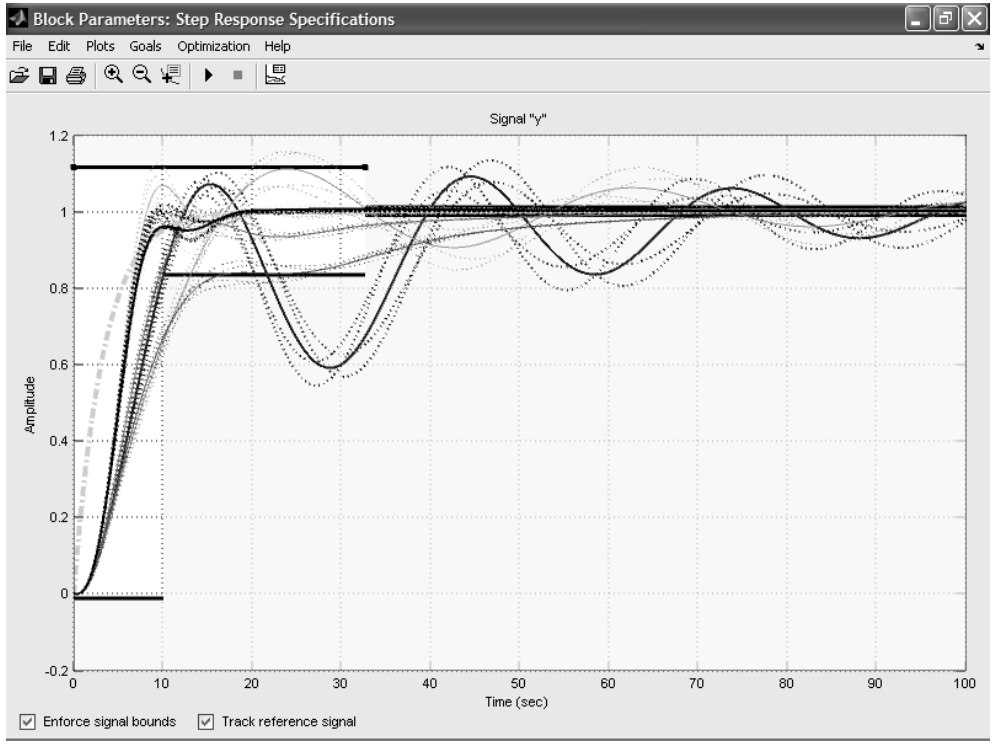


*Рис. 10.16. Окно **Optimization Progress** примера PID Tracking Demo*

10.3.3. Оптимизация системы магнитной «левитации» стального шарика

С помощью электромагнита, питаемого током, можно поднять шарик с поверхности и удерживать его в пространстве. Для этого нужна простая система регулирования, ослабляющая ток в электромагните, если шарик поднимается выше некоторого порогового уровня. Поскольку шарик имеет вес, то такая система оказывается инерционной и для нее характерен колебательный процесс. Задача оптимизации заключается в подборе таких параметров PID-регулятора, входящего в систему регулирования (рис. 10.18), при которых переходный процесс подъема шарика укладывается в заданный интервал времени и имеет близкий к монотонному характер.

На рис. 10.18 снизу показаны установки в окне параметров PID-регулятора и осциллограммы тока до проведения оптимизации системы регулирования. Не-

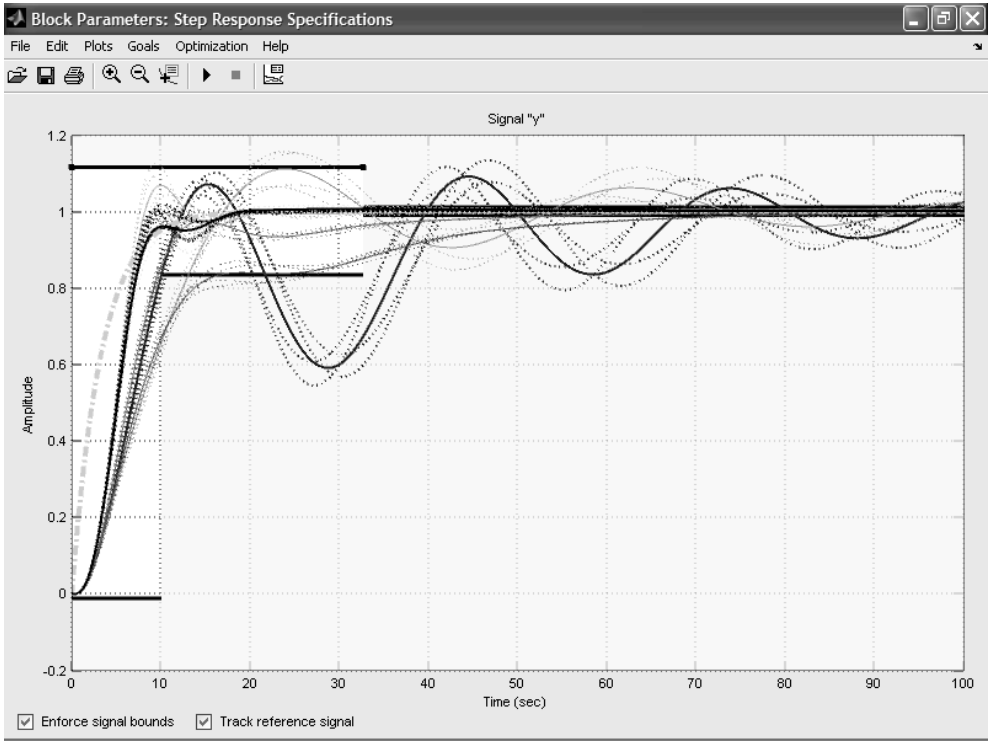


Рис. 10.17. Окно контроля и норм допуска переходных процессов примера PID Tracking Demo

трудно заметить, что она имеет довольно сложный вид – в виде импульсов с монотонно спадающей амплитудой. Это указывает на явно релаксационный характер работы системы регулирования.

Для проведения оптимизации достаточно активизировать субмодель с именем Magnetic Levitation Plant Model. Появится субмодель, диаграмма которой представлена на рис. 10.19. На ней представлены субмодели цифро-аналогового D/A и аналого-цифрового A/D преобразователей. С остальными субмоделями можно ознакомиться, открыв их окна. В частности, важное значение имеет субблок **Limit**, задающий нелинейность данной субмодели.

Для запуска режима оптимизации достаточно активизировать блок **Position Constrained** – он выделен зеленым цветом фона. Появится окно вывода переходных процессов и задания допусков на них – рис. 10.20. В нем можно мышью переместить линии, задающие допуски. После пуска оптимизации (кнопкой с изображением треугольника в панели инструментов) можно наблюдать построение переходных процессов (окно **Optimization Progress** не представлено).

В этом примере уже после двух итераций переходный процесс при пуске «левитации» приобрел монотонный характер, что видно из рис. 10.20 и осциллограммы переходного процесса, показанного на рис. 10.21. Этот пример наглядно показывает, сколь сильно отличается переходный процесс в оптимизированной системе от начального переходного процесса при отсутствии оптимизации. Это важный фактор при проектировании сложных систем управления.

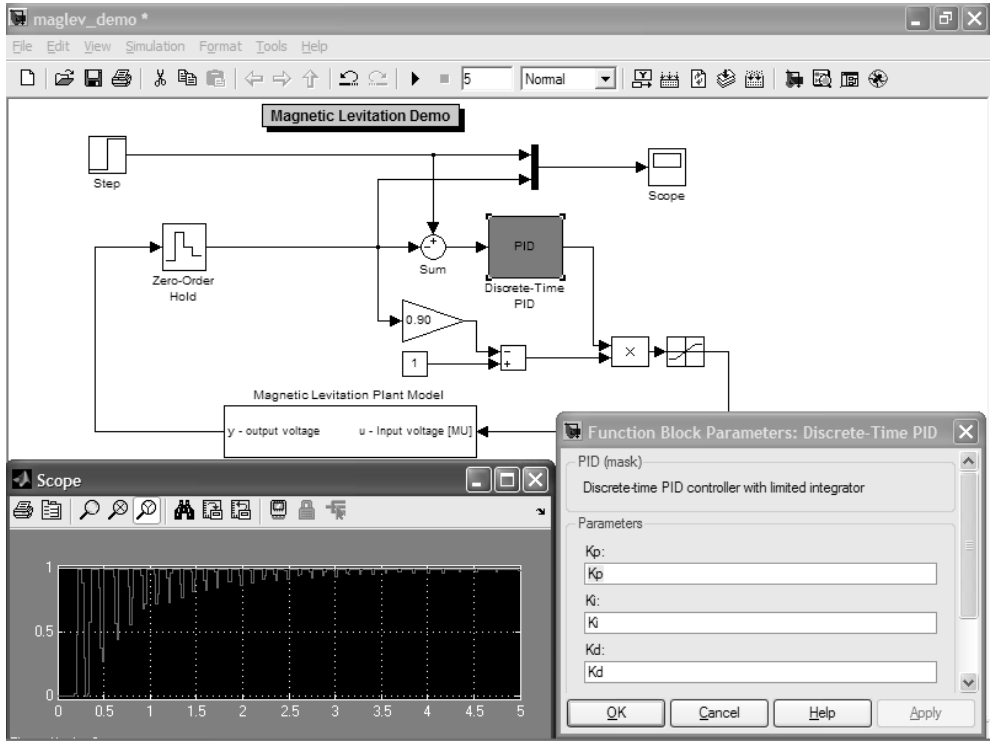


Рис. 10.18. Окно диаграммы модели «левитации» стального шарика

10.3.4. Оптимизация системы энергетического преобразователя

На рис. 10.22 показана еще одна сложная система, которая нуждается в оптимизации. Это энергетический преобразователь, содержащий трехфазный тиристорный управляемый выпрямитель (конвертор), формирующий пульсирующий постоянный ток для питания двигателя постоянного тока (он представлен соответствующим эквивалентом). Для управления конвертором используется шестифазный синхронный генератор импульсов. Задача оптимизации заключается в подборе величин Kp и Ku PI регулятора тока, с тем чтобы обеспечить приемлемый ход переходных процессов при изменении работы системы.

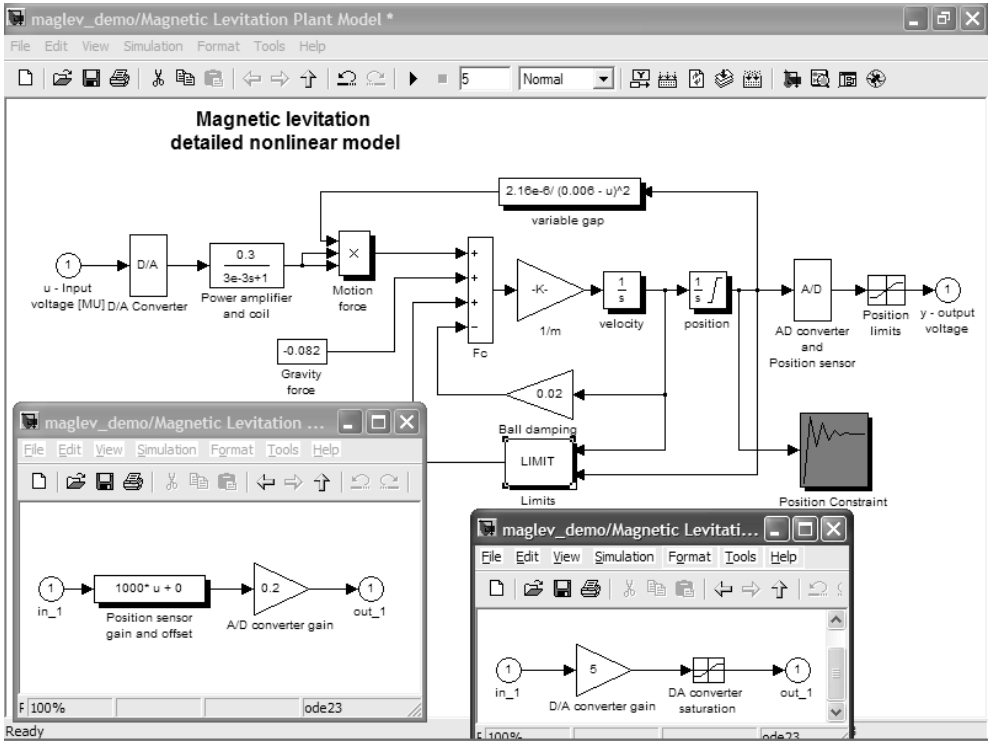


Рис. 10.19. Окно диаграммы субмодели Magnetic Levitation Plant Model

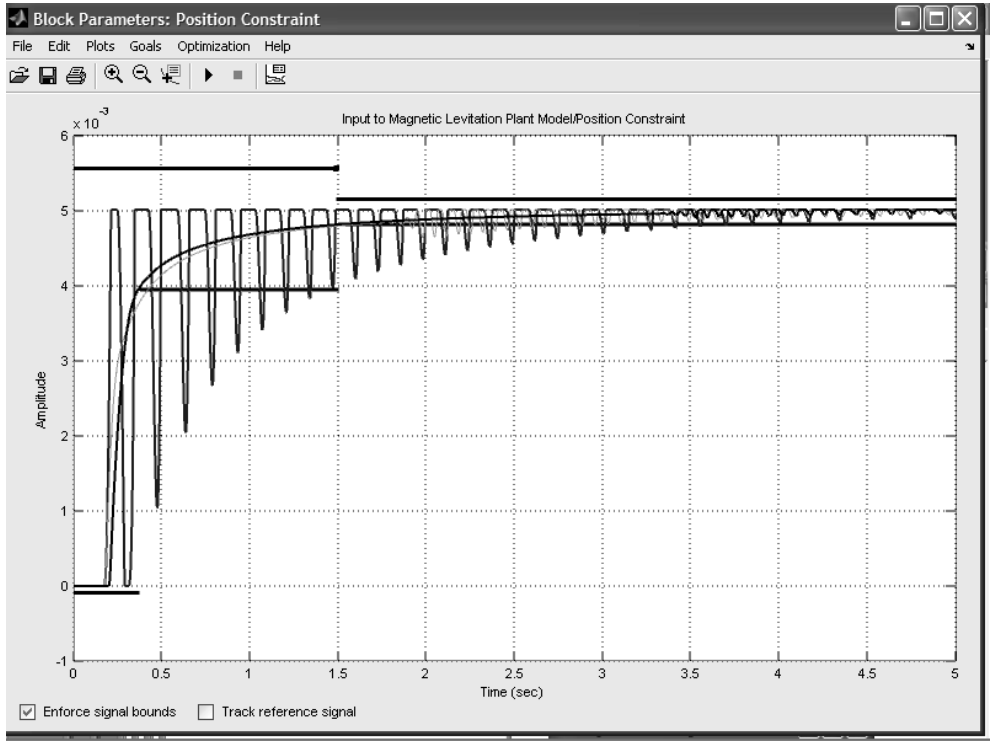


Рис. 10.20. Окно вывода переходных процессов субмодели Magnetic Levitation Plant Model

Диаграмма модели энергетического преобразователя показана на рис. 10.22. В диаграмме используются блоки как пакета Simulink, так и пакета SimPowerSystems, используемого для построения моделей энергетических систем (этот пакет расширения подробно описан в следующем уроке). Благодаря применению PI-регулятора система имеет замкнутый характер, и ее поведение сильно зависит от параметров блоков Kp и Ku. Цель регулирования заключается в отслеживании создаваемого преобразователем тока с опорным током Id_ref, создаваемым блоком Reference Current Id_ref.

Система настроена так, что идеальным были бы уровни тока двигателя с момента пуска, равные 10 и 25 А (см. нижние осциллограммы работы системы до оптимизации, показанные на рис. 10.23). Именно они задаются блоком **Reference Current Id_ref**. Нетрудно заметить, что эта цель без оптимизации не достигается. Средние уровни пульсирующего тока явно не дотягивают до нужных уровней, а время переходных процессов значительно.

Для запуска оптимизации надо активизировать блок **Current Regulation Specs**. Он запускает MATLAB-программу, начинающую процесс оптимизации, который заключается в итерационном изменении Kp и Ku до тех пор, пока цели

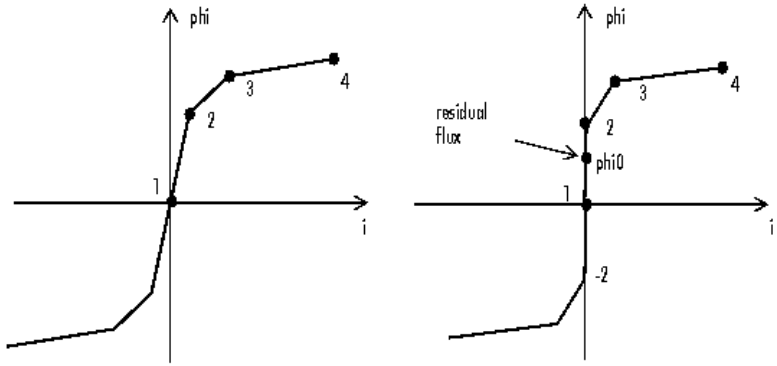


Рис. 10.21. Оциллограмма переходного процесса после проведения оптимизации

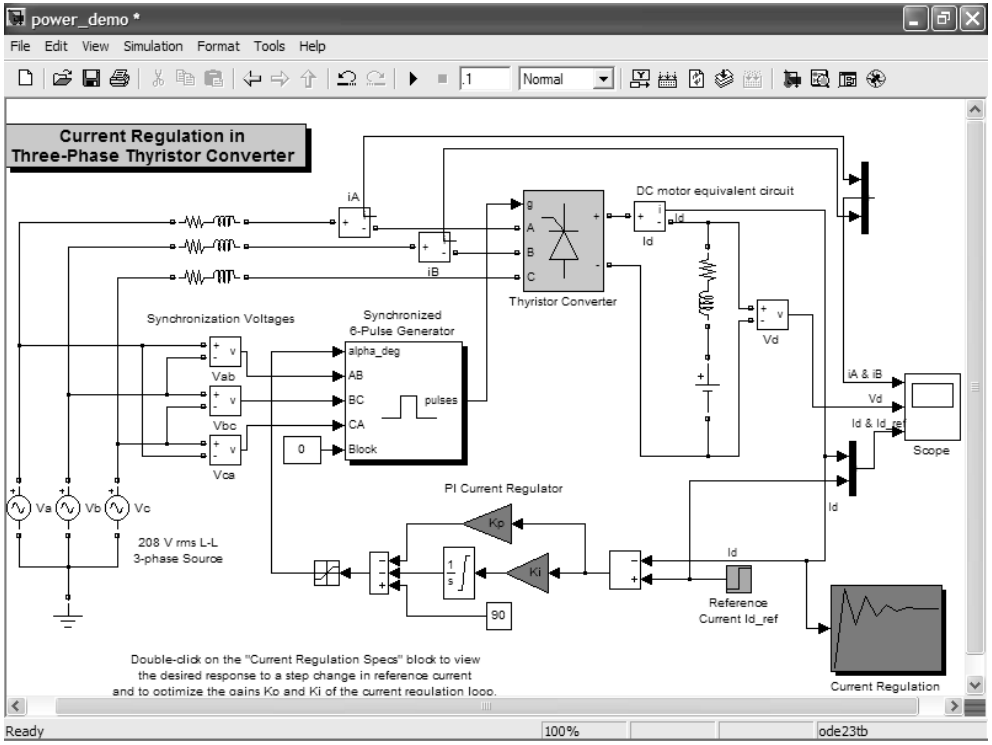


Рис. 10.22. Диаграмма модели системы энергетического преобразователя

Рис. 10.23. Оциллограммы работы модели энергетического преобразователя до оптимизации

оптимизации не будут достигнуты. При этом открывается окно **Optimization Progress**, показанное на рис. 10.24. Из него видно, что оптимизация завершается за 4 итерации и ведет к выдаче оптимальных значений Kp и Ku.

На рис. 10.25 показано окно параметров блока **Current Regulation Specs**, содержащее граничные линии установки допусков переходных процессов и сами переходные процессы, полученные в ходе итераций оптимизации. Нетрудно заметить, что после оптимизации переходные процессы вошли в заданную норму.

Переходные процессы после оптимизации представлены в виде осциллограмм на рис. 10.26. Сравнение их с осциллограммами рис. 10.23 показывает, что основные цели оптимизации явно достигнуты, хотя и неидеально. В частности, хорошо видно, что из-за инерционности системы управления высокочастотные пульсации тока практически не сглажены – однако это совершенно не влияет на работу инерционного двигателя постоянного тока. Уровень тока в 10 А достигнут как среднее значение пульсирующего тока, а уровень 25 А немного не достигнут. Время переходных процессов существенно уменьшилось.

10.3.5. Функции пакета расширения Simulink Response Optimization

Для подготовки управляющих работой Simulink-моделей MATLAB-программ пакет Simulink Response Optimization имеет несколько фукций. Ниже они приведены по категориям.

Оптимизация отклика системы:

- `getsro` – получает проект оптимизации отклика для данной Simulink-модели;
- `ncdupdate` – модернизирует модели со старыми блоками **NCD**;
- `newsro` – создает Simulink-проект оптимизации отклика по умолчанию;
- `optimize` – запускает оптимизацию отклика.

Ограничения и параметры:

- `findconstr` – находит ограничения на данный отклик;
- `findpar` – находит спецификации для данного параметра;

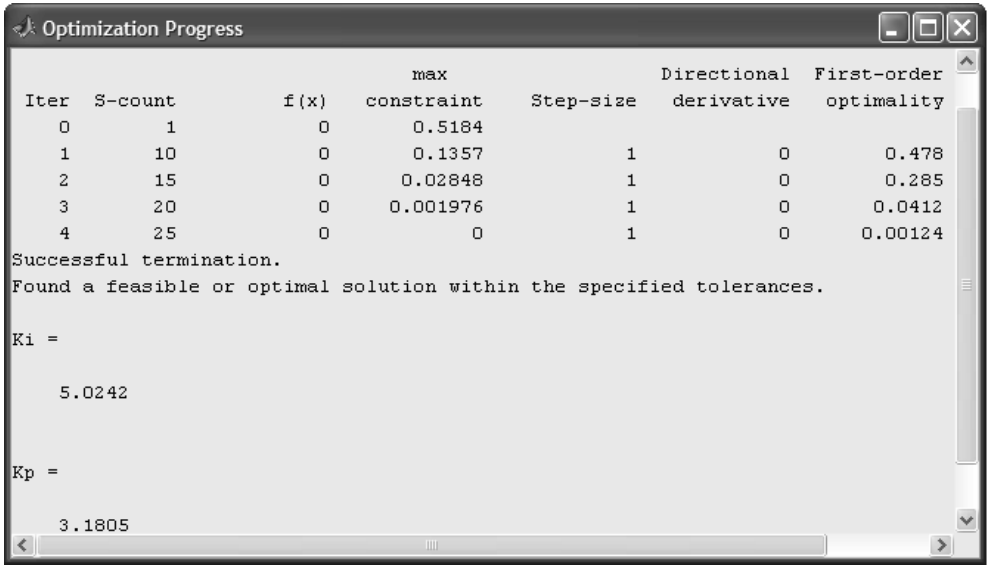


*Рис. 10.24. Окно **Optimization Progress** для примера на оптимизацию*

- `gridunc` – строит N-D-сетку из параметров сомнительной ценности;
- `initpar` – задает начальные значения параметров;
- `randunc` – задает случайные значения параметров;
- `setunc` – устанавливает параметры сомнительной ценности в процессе оптимизации.

Оптимизация и установки моделирования:

- `optimget` – восстанавливает поток оптимизатора;
- `optimset` – изменяет назначение оптимизатора;
- `simget` – восстанавливает текущие назначения моделирования;
- `simset` – именяет назначение моделирования.

С детальным описанием этих функций и правилами их записи можно познакомиться по справке пакета расширения Simulink Response Optimization.

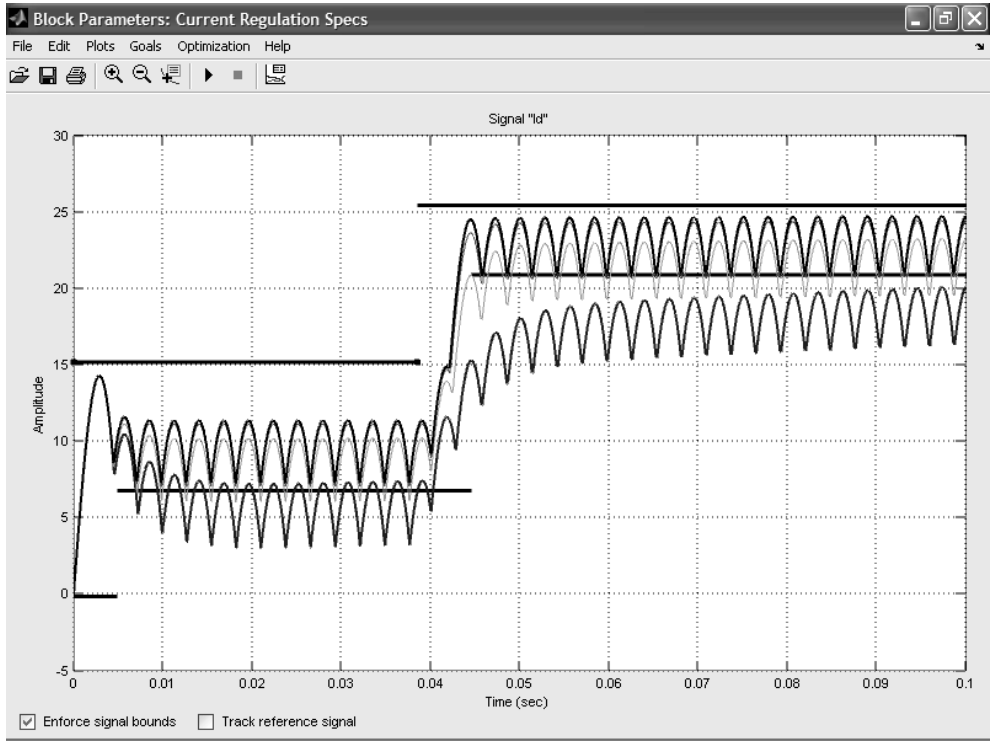


Рис. 10.25. Переходные процессы в ходе оптимизации

Рис. 10.26. Оциллограммы работы модели энергетического преобразователя после оптимизации

Урок 11

Моделирование в электроэнергетике

Simulink предназначен для моделирования устройств и систем самого общего характера. Однако важное место занимает моделирование в ряде специальных областей науки и техники. К таким областям относится электроэнергетика. В данном уроке мы познакомимся с одним из самых мощных пакетов моделирования электроэнергетических и электротехнических устройств – SimPowerSystems.

11.1. Пакет расширения SimPowerSystems

11.1.1. Назначение пакета расширения SimPowerSystems

Для моделирования энергетических систем, включая устройства электротехники и промышленной электроники, в прежних версиях MATLAB служил пакет расширения Power System Blockset, реализации которого были описаны в [5, 6, 13, 16, 32, 39]. Теоретические основы математического моделирования таких устройств даны в [71–81]. В версиях MATLAB 6.5 SP1/7 + Simulink 5/6 и выше пакет переименован в SimPowerSystems Blockset.

Описание пакета SimPowerSystems Blockset в формате PDF занимает около 650 страниц. Ниже мы ограничимся гораздо более компактным описанием. Доступ к пакету возможен из окна вьювера библиотек и показан на рис. 11.1.

11.1.2. Состав библиотек SimPowerSystems Blockset

В этот пакет (версии 3 и 4) входят библиотеки следующего назначения:

- **Extra Library** – специальная расширенная библиотека блоков-масок;
- **Application Librares** – библиотека применений (новый раздел в версии пакета 4);
- **Electrical Sources** – источники электрической энергии и сигналов;
- **Elements** – линейные и нелинейные компоненты электротехнических и электронных устройств;
- **Machines** – электрические машины;
- **Measurements** – измерительные и контрольные устройства;
- **Phasor Element** – элемент моделирования трехфазных систем;
- **Power Electronics** – блоки устройств энергетической электроники;
- **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.

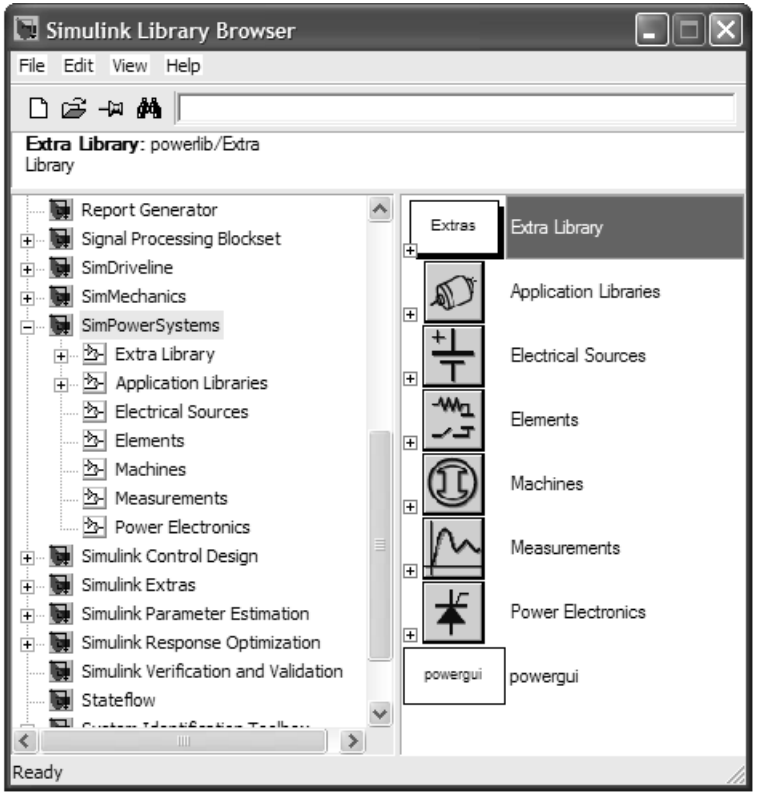


Рис. 11.1. Доступ к библиотеке блоков пакета SimPowerSystems Blockset

11.1.3. Параметры и единицы их измерения

При указании параметров моделей компонентов и физических единиц для их измерения используются следующие обозначения (в скобках даны обозначения, принятые в отечественной литературе):

Параметр	Единица измерения	Сокращенное обозначение
Time (время)	Second	s (с)
Length (длина)	Meter	m (м)
Mass (масса)	Kilogram	kg (кг)
Energy (энергия)	Joule	J (Дж)
Current (ток)	Ampere	A (А)
Voltage (напряжение)	Volt	V (В)
Frequency (частота)	Hertz	Hz (Гц)

Параметр	Единица измерения	Сокращенное обозначение
Active power (активная мощность)	Watt	W (Вт)
Apparent power (полная мощность)	Volt ampere	VA (ВА)
Reactive power (реактивная мощность)	Volt-ampere reactive	var (вар)
Impedance (импеданс)	Ohm	Ω (Ом)
Resistance (сопротивление)	Ohm	Ω (Ом)
Inductance (индуктивность)	Henry	H (Гн)
Capacitance (емкость)	Farad	F (Ф)
Flux linkage (поток сцепления)	volt second	V·S (В·с)
Rotation speed (скорость вращения)	radians per second	rad/s (рад/с)
	revolutions per minute	rpm (оборот./мин)
Torque (вращающий момент)	newton meter	N·m (Н·м)
Inertia (инерция)	kilogram (meter)^2	kg·m^2 (кг·м^2)
Friction factor (коэффициент трения)	newton meter second	N·m·s (Н·м·с)

В этой таблице приведены лишь основные параметры и единицы их измерения. Некоторые из параметров будут рассмотрены по мере описания моделей.

11.2. Источники электрической энергии и их применение

11.2.1. Типы источников электрической энергии

Источники электрической энергии являются первичными компонентами энергетических систем и устройств. Большинство электротехнических устройств являются 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. 11.2).

В нем представлены семь 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;
- **Three Phase Programmable Voltage Source** – программируемый источник трехфазного напряжения;
- **Three Phase Source** – источник трехфазного напряжения.

Эти источники образуют функционально полный набор источников электрической энергии и минимальный набор источников сигналов (в других расширениях MATLAB, кроме источников синусоидальных сигналов, можно задавать великое множество источников сигналов самой разной формы). Данных наборов

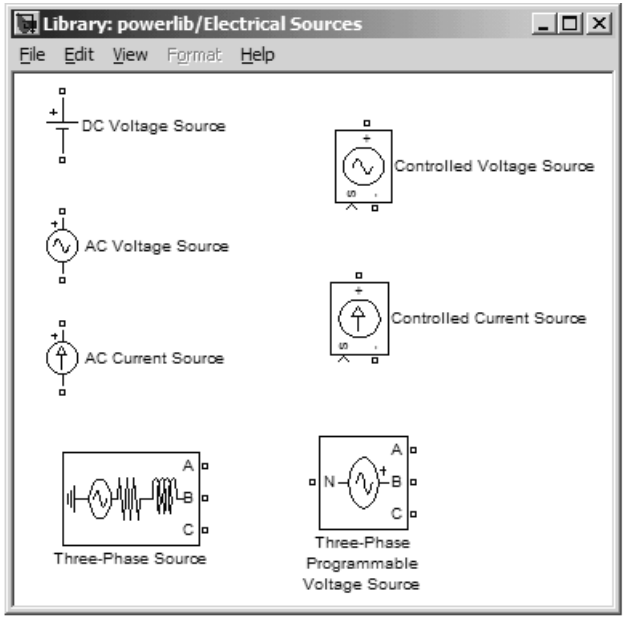


Рис. 11.2. Окно библиотеки источников Electrical Sources

вполне достаточно для проектирования энергетических устройств. Назначение и роль первых трех источников в особых комментариях не нуждаются.

11.2.2. Пример применения источника постоянного тока

На рис. 11.3 представлены вид блока и окно установки параметров источника напряжения постоянного тока **DC Voltage Source**. Основным его параметром является уровень (амплитуда Amplitude) напряжения, по умолчанию равный 100 В. Обратите внимание на список **Measurements (Инструменты)**. Он присутствует во многих других блоках и служит для создания дополнительного выхода для сигнала, контролирующего блок. Например, для блока постоянного напряжения этот сигнал дает значение напряжения источника. При этом список дает выбор между Voltage и None (отсутствие контроля, параметр, заданный по умолчанию).

11.2.3. Пример применения управляемого источника 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 задается только начальный ток (по умолчанию рав-

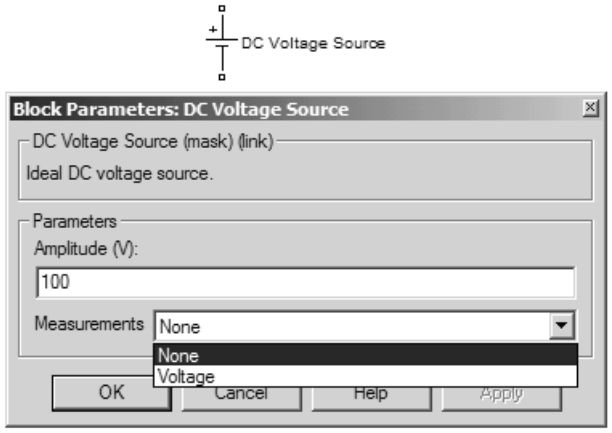


Рис. 11.3. Источник постоянного напряжения и окно установки его параметров

ный 0). При этом временная зависимость выходного тока определяется временной зависимостью управляющего тока. 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. Управляемые источники напряжения и тока позволяют моделировать такой важный класс электрических цепей, как параметрические цепи (цепи, параметры которых зависят от времени).

11.2.4. Примеры применения источника переменного тока

Идеальный (с бесконечно большим внутренним сопротивлением) источник переменного тока с заданной амплитудой, частотой и фазой (**AC Current Source**) задает ток, который меняется по синусоидальному закону и описывается выражением

$I = \text{Amplitude} \times \sin(2\text{pi} \times \text{Frequency} \times t + \text{Phase} \times \text{pi}/180)$.

Соответственно, идеальный (с нулевым внутренним сопротивлением) источник переменного напряжения с заданной амплитудой, частотой и фазой (**AC Voltage Source**) задает напряжение, меняющееся по синусоидальному закону и описываемое выражением

$U = \text{Amplitude} \times \sin(2\text{pi} \times \text{Frequency} \times t + \text{Phase} \times \text{pi}/180)$.

Характеризующие эти источники параметры общеизвестны и задаются в окнах параметров.

На рис. 11.4 представлена модель суммирования на нагрузке сигналов от двух источников переменного напряжения с разными амплитудами и частотами. Сигнал в нагрузке с первого взгляда напоминает амплитудно-модулированную си-

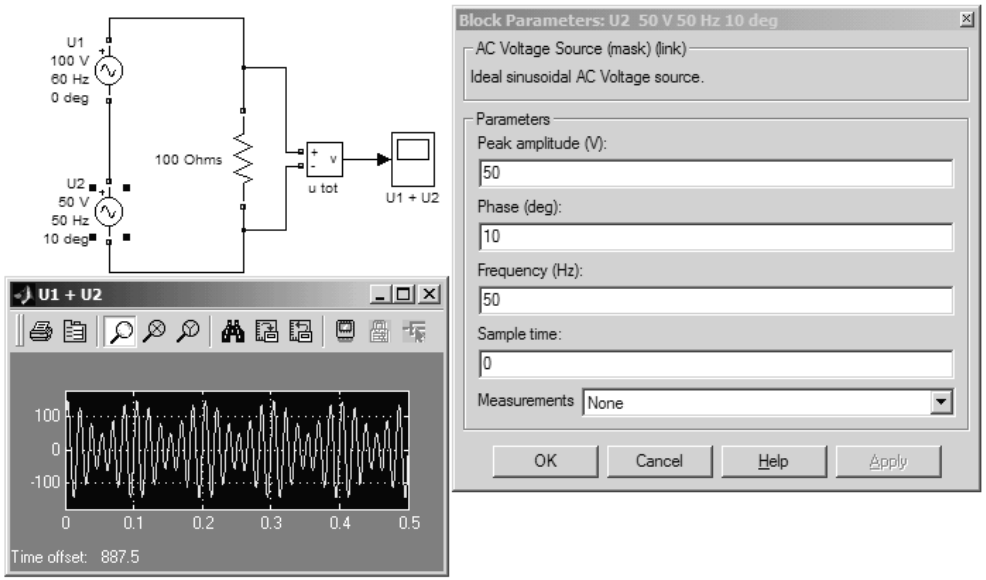


Рис. 11.4. Модель суммирования двух источников переменного напряжения и окно установки параметров одного из источников

нусоиду, но на самом деле эти типичные биения – изменение амплитуды двухкомпонентного сигнала с разностной частотой. На этом рисунке показано окно установки параметров источника переменного напряжения. Установки в нем очевидны.

Чтобы измерить напряжение на резисторе, используется блок `u tot` из библиотеки измерительных блоков Measurements. Мы рассмотрим эту библиотеку немного позже.

Аналогичный представленному на рис. 11.4 результат можно получить, суммируя токи от двух источников переменного тока. Такую модель демонстрирует рис. 11.5. Для контроля тока в нагрузке используется блок `i tot` из библиотеки измерительных блоков Measurements.

В дальнейшем наличие списка и установки общеизвестных параметров, такие как амплитуда, частота, фаза, резистивность, емкость или индуктивность, особо оговариваться не будут ввиду их очевидности для пользователя, знакомого с электротехникой. Другие пользователи данный пакет применять просто не будут.

11.2.5. Моделирование амплитудной модуляции

Для получения амплитудно-модулированного сигнала можно использовать модель, представленную на рис. 11.6. Собственно амплитудно-модулированный сигнал получается умножением высокочастотного сигнала с частотой 60 Гц на низкочастотный сигнал с частотой 5 Гц с добавленной постоянной составляющей.

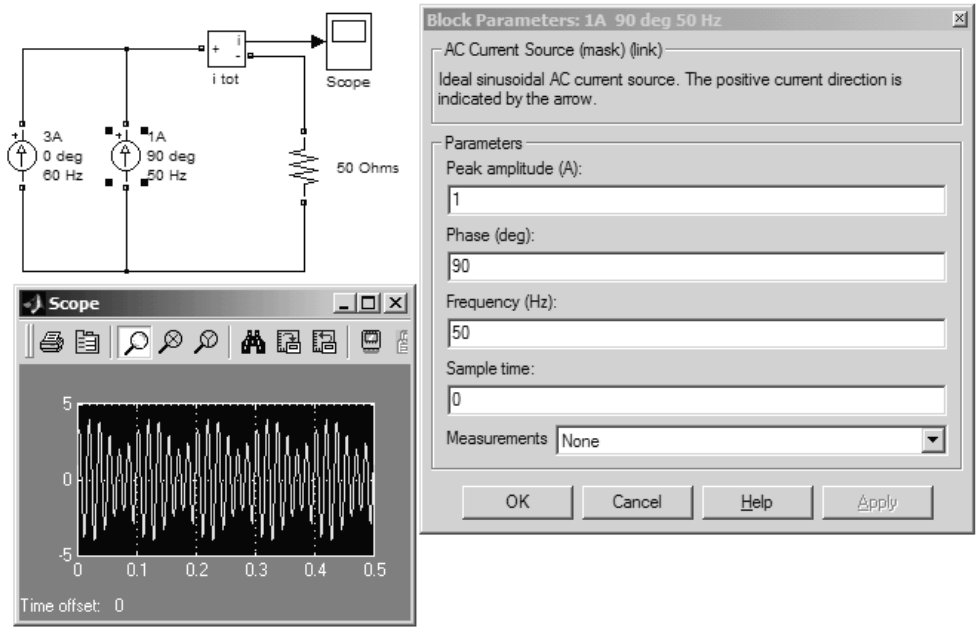


Рис. 11.5. Модель суммирования двух источников переменного тока и окно установки параметров одного из источников

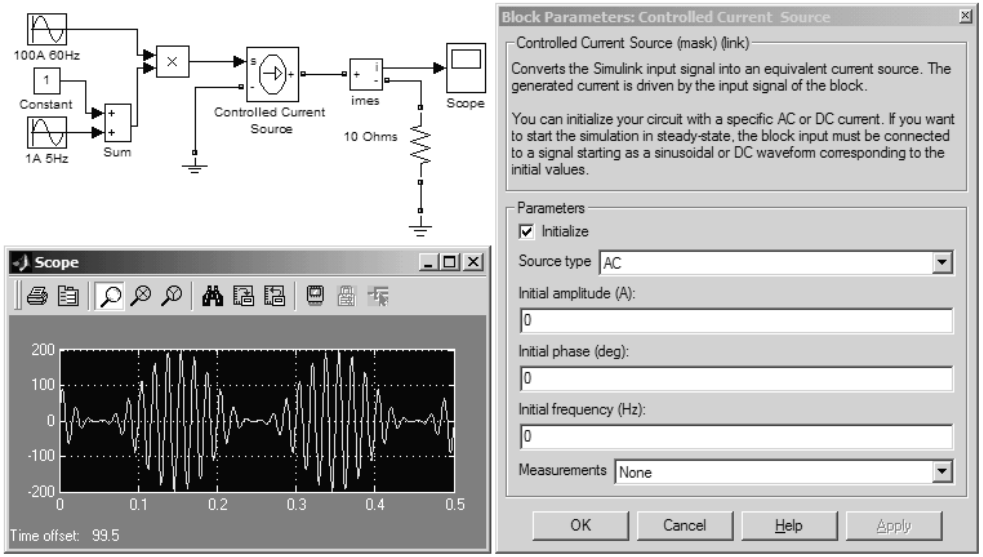


Рис. 11.6. Модель для создания амплитудно-модулированного тока

Результирующий сигнал подается на управляемый источник тока и через блок контроля тока – в резистивную нагрузку.

Представленный на рис. 11.6 пример соответствует 100% модуляции. Виртуальный осциллограф подключен к контрольному выходу блока контроля тока. Как задаются резисторы в этих моделях, станет ясно чуть ниже.

11.3. Основные элементы электротехнических устройств и систем

11.3.1. Библиотека компонентов Elements

Основная библиотека компонентов (рис. 11.7) содержит ряд блоков, имеющих достаточно универсальный характер. С помощью одного такого блока можно, как правило, создать блоки нескольких простых компонентов и учитывать различные паразитные их параметры.

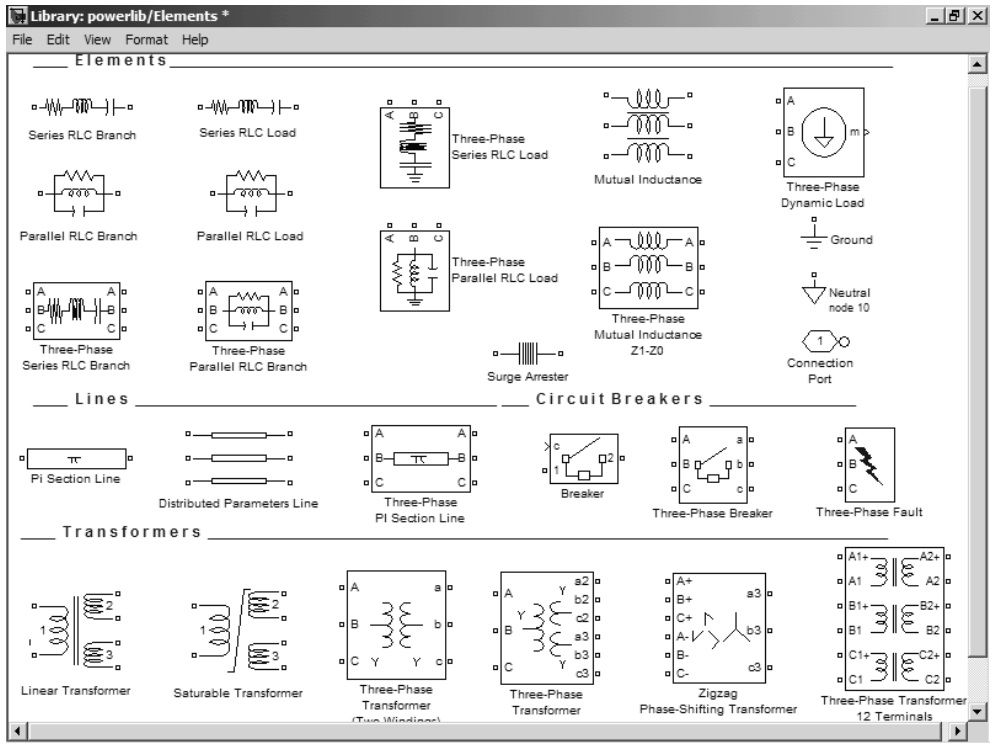


Рис. 11.7. Окно библиотеки компонентов Elements

Эта библиотека содержит четыре раздела:

- `Elements` – элементы электротехнических устройств;
- `Lines` – линии передачи;
- `Circuis Breakers` – выключатели;
- `Transformers` – трансформаторы.

Раздел `Elements` содержит полтора десятка характерных компонентов электрических устройств:

- **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;
- **Three-Phase Series RLC Branch** – трехфазная последовательная RLC-цепь;
- **Three-Phase Series RLC Load** – трехфазная последовательная RLC-цепь с нагрузкой;
- **Three-Phase Parallel RLC Branch** – трехфазная параллельная RLC-цепь;
- **Three-Phase Parallel RLC Load** – трехфазная параллельная RLC-цепь с нагрузкой;
- **Mutual Inductance** – блок взаимной индуктивности;
- **Three-Phase Mutual Inductance Z1-Z0** – блок взаимной индуктивности трехфазный;
- **Three-Phase Dynamic Load** – трехфазная динамическая нагрузка;
- **Surge Arrester** – ограничитель пиковых напряжений;
- **Ground** – земля;
- **Neutral Note 10** – нейтраль;
- **Connection port** – порт подключения.

Раздел линий передачи **Lines** содержит следующие блоки:

- **PI Section Line** – линия с сосредоточенными параметрами;
- **Distributed Parameters Line** – линия с распределенными параметрами;
- **Three-Phase PI Section Line** – трехфазная линия с сосредоточенными параметрами.

В разделе **Circuis Breakers** имеются блоки выключателей:

- **Breaker** – выключатель управляемый;
- **Three-Phase Breaker** – трехфазный выключатель управляемый;
- **Three-Phase Fault** – трехфазный разрядник (замыкатель фаз на землю).

Раздел трансформаторов содержит такие блоки, как:

- **Linear Transformer** – линейный трансформатор;
- **Saturable Transformer** – нелинейный трансформатор;
- **Mutual Inductance** – блок взаимной индуктивности однофазный;
- **Three-Phase Transformer (Three Winding)** – трансформатор с тремя обмотками, имеющими отводы от их середины;
- **Zigzag Phase-Sifting Transformer** – трехфазный фазосдвигающий трансформатор;
- **Three-Phase Mutual Inductance** – блок взаимной индуктивности трехфазный;

- **Three-Phase Transformer 12 Terminal** – трехфазный блок из трех однофазных трансформаторов, имеющий 12 портов.

11.3.2. Примеры моделирования RLC-цепей

В состав библиотеки Elements входят две последовательные и две параллельные *RLC*-цепи и их трехфазные варианты. 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 параллельной **Parallel RLC Load**) дополнительно задаются допустимые мощности рассеяния: активная для резистора и реактивные для индуктивности и конденсатора. Последовательные и параллельные RLC-цепи могут использоваться для моделирования колебательных контуров и создания эквивалентов нагрузки.

Для ввода отдельных элементов (резистора **R**, конденсатора **C** и индуктивности **L**) можно использовать любую из RLC-цепей, задав параметрам значения, соответствующие отсутствию ненужных компонентов, – рис. 11.8. 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 в трехфазных цепях, **RsCs**-цепи в моделях ключей (они будут описаны ниже).

Благодаря этому правилу число простых моделей в пакете SimPowerSystems Blockset сокращено. Кроме того, это правило позволяет быстро модернизировать

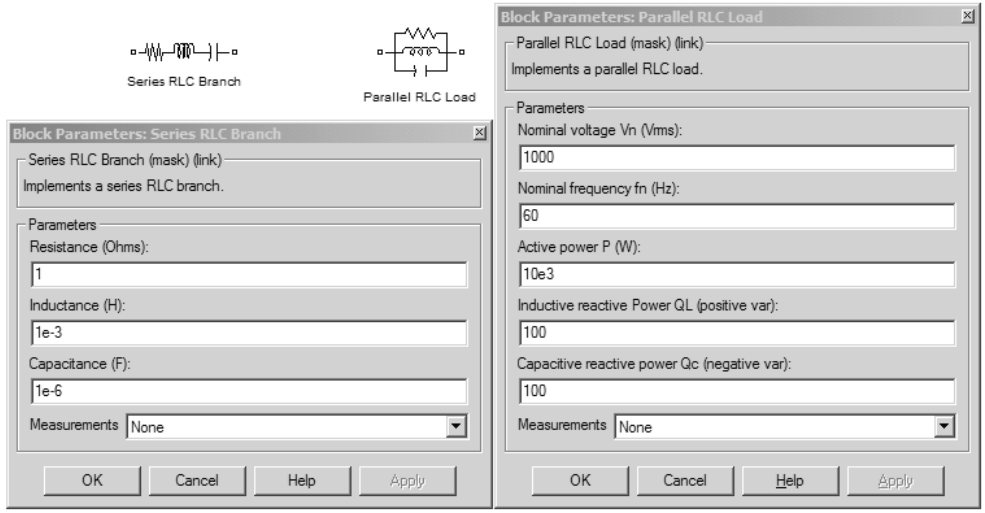


Рис. 11.8. Окна установки параметров RLC-цепей

отдельные цепи, например превращать резистор **R** в **RL**- или **RLC**-цепь, не вводя новых компонентов в уже составленную схему, а просто задав их в окне параметров RLC-цепей. При этом компоненты L и C могут представлять паразитные индуктивности и емкости резисторов.

Рисунок 11.9 показывает пример моделирования простой цепи, содержащей резистор, параллельно которому включена последовательная R1C1-цепь. Эта комбинированная цепь питается от источника переменного тока, а напряжение на ней контролируется осциллографом. Как и следовало ожидать, присутствует фазовый сдвиг, обусловленный конденсатором R1C1-цепи.

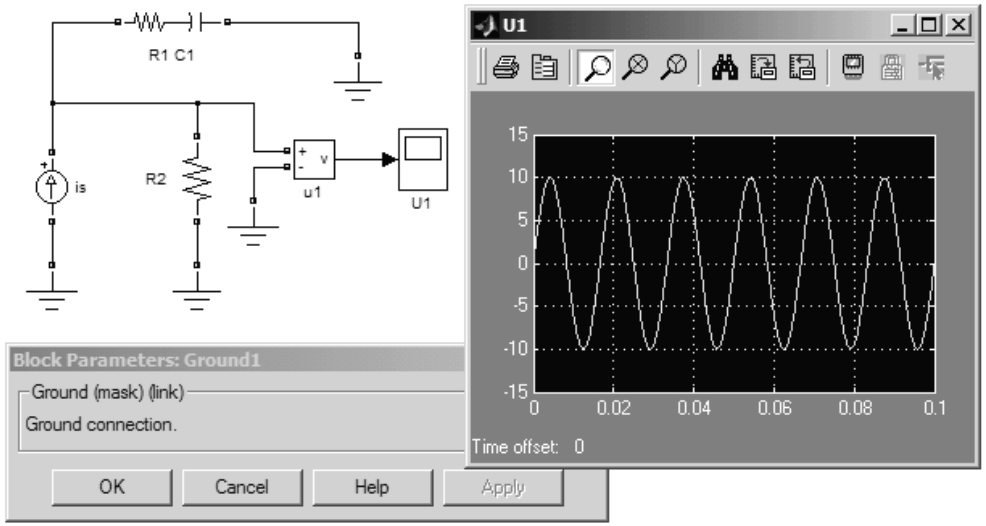


Рис. 11.9. Пример моделирования простой цепи

В следующем примере (рис. 11.10) рассматривается сложный делитель напряжения. Он содержит резистор R3 и сложную цепь, содержащую последовательную R1C1L1-цепочку, параллельно которой включены резистор R2 и конденсатор C2. Для объединения трех последних компонентов используется блок `Junction Point`.

Рисунок 11.11 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. Обратите внимание на окно параметров последовательной RLC-цепи.

Блок **Powergui** обеспечивает графический интерфейс пользователя и контроль за параметрами модели в целом. В следующем разделе мы рассмотрим применение этого блока более подробно.

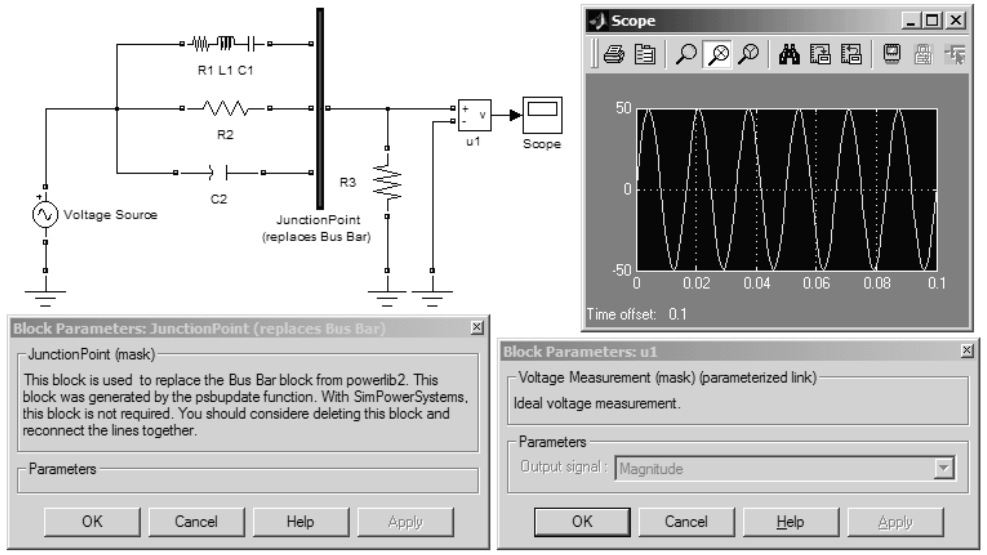


Рис. 11.10. Пример моделирования сложного делителя напряжения

11.3.3. Работа с блоком Powergui

Библиотека пакета SimPowerSystems Blockset симеет особый блок `Powergui` – см. рис. 11.1. Этот блок не имеет ни входов, ни выходов и может включаться в состав любой модели как блок вызова графического интерфейса пользователя. В блок поступают все данные о модели и результатах моделирования. Возможности блока зависят от модели, в которой блок используется. Рассмотрим их на примере модели рис. 11.12.

Активизация пиктограммы **Powergui** выводит окно интерфейса (оно показано на рис. 11.31 справа). Окно **powergui** позволяет контролировать состояние переменных модели в момент инициализации и после моделирования (рис. 11.12), обращаться к LTI-вьюверу, выполнять быстрое преобразование Фурье и т. д.

Рисунок 11.13 показывает применение LTI-вьювера. Для этого достаточно активизировать команду `Use LTI Viewer` на панели GUI-интерфейса. В появившемся окне (на рис. 11.13 оно показано в центре) надо выбрать переменные, которые будут использованы для анализа. Используя в нем кнопку **Open LTI-вьювер**, можно открыть окно LTI-вьювера, показанное на рис. 11.13 слева. В нем строятся возможные для данной модели переходные характеристики по току и напряжению.

С помощью блока можно осуществлять много полезных операций. Для этого совместно с ним надо использовать блоки измерений. Например, если использовать блок `Impedance Measurement`, то можно получить графики модуля импеданса и фазового сдвига. А применение блока **Fourier** позволяет выполнить быстрое преобразование Фурье и получить спектрограмму.

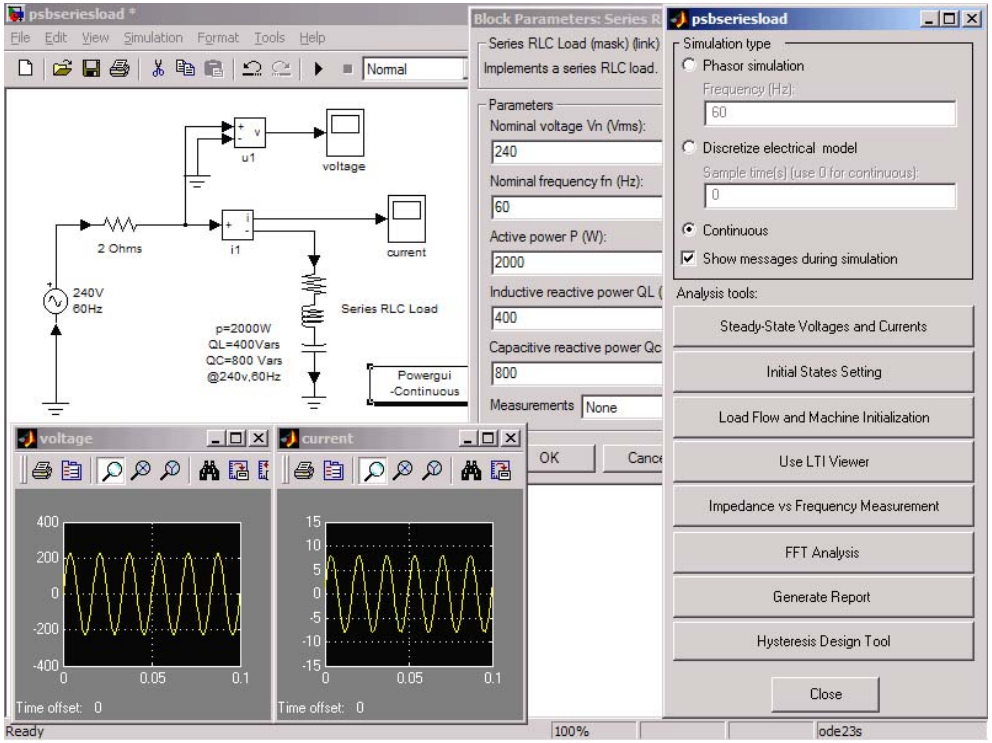


Рис. 11.11. Пример моделирования нагрузочной последовательной RLC-цепи, подключенной к источнику переменного напряжения

Рисунок 11.14 демонстрирует получение характеристик импеданса для последовательного *RLC* колебательного контура. На контур подается сумма сигналов с частотами 60 и 300 Гц, контур настроен на пятую гармонику сигнала с частотой 60 Гц, создаваемую источником переменного тока Isource. Осциллограммы показывают временные зависимости напряжения на контуре и тока через него. А с помощью блока `Powergui`, переименованного в `Continuous`, можно вывести окно блока и, активизируя кнопку `Impedance vs Frequence Measurement`, построить частотную характеристику модуля импеданса и его фазы.

Блок `Powergui` позволяет также подготовить краткий отчет по моделированию для текущей модели. Для этого достаточно исполнить команду `Generate Report` (**Генерация отчета**). Появится окно генерации отчета. Установив нужные данные для отчета, надо нажать кнопку **Generate Report** уже в этом окне. Появится обычное окно для записи отчета. Отчет записывается в виде файла *.rep.

Иные возможности блока `Powergui` мы рассмотрим по мере описания других моделей.

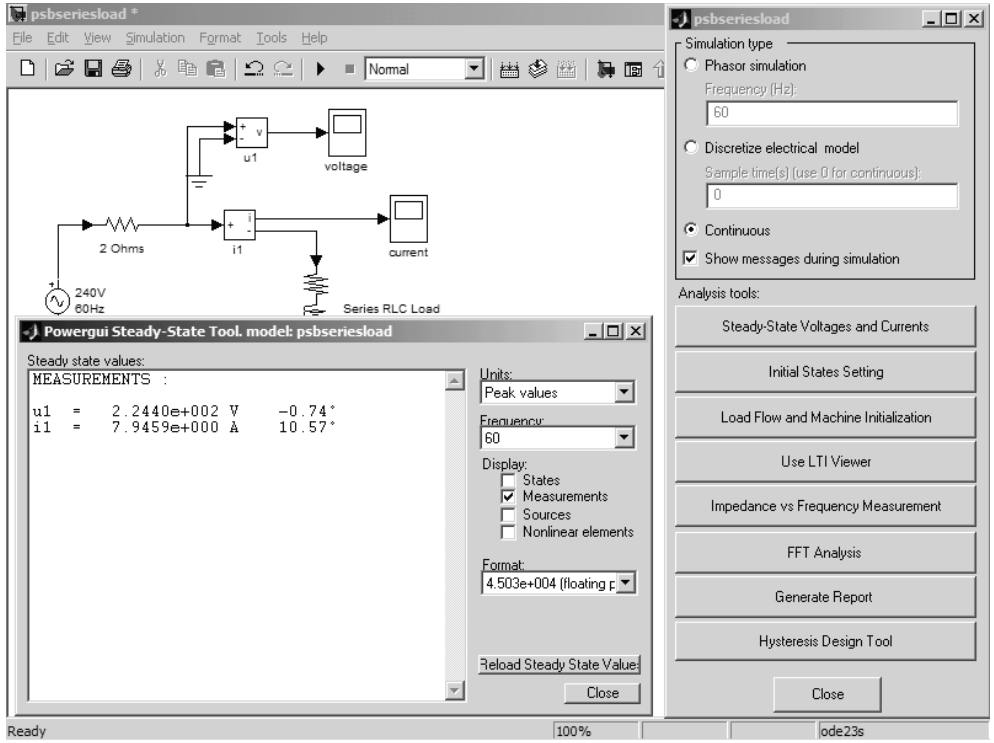


Рис. 11.12. Контроль состояния переменных после моделирования

11.3.4. Моделирование устройств с однофазными трансформаторами

В пакете SimPowerSystems Blockset имеется блок линейного трансформатора **Linear Transformer** – рис. 11.15. Он задается индуктивностью $\mathbf{L_m}$ и сопротивлением потерь в сердечнике $\mathbf{R_m}$ первичной обмотки трансформатора, а также омическими сопротивлениями $\mathbf{R_i}$ и индуктивностями рассеяния $\mathbf{L_i}$ всех обмоток трансформатора (i = 1, 2, 3). Полезно обратить внимание на то, что некоторые параметры трансформатора задаются списками, поскольку трансформатор может иметь несколько обмоток.

Пример моделирования системы с линейным трансформатором дан на рис. 11.16. В модели используется блок трехобмоточного трансформатора с первичной обмоткой, подключенной к источнику переменного напряжения. Каждая из двух вторичных обмоток имеет свою нагрузку, есть и общая нагрузка. Нагрузка к нижней вторичной обмотке отключается выключателем Breaker, что создает переходный процесс, хорошо иллюстрированный осциллограммами.

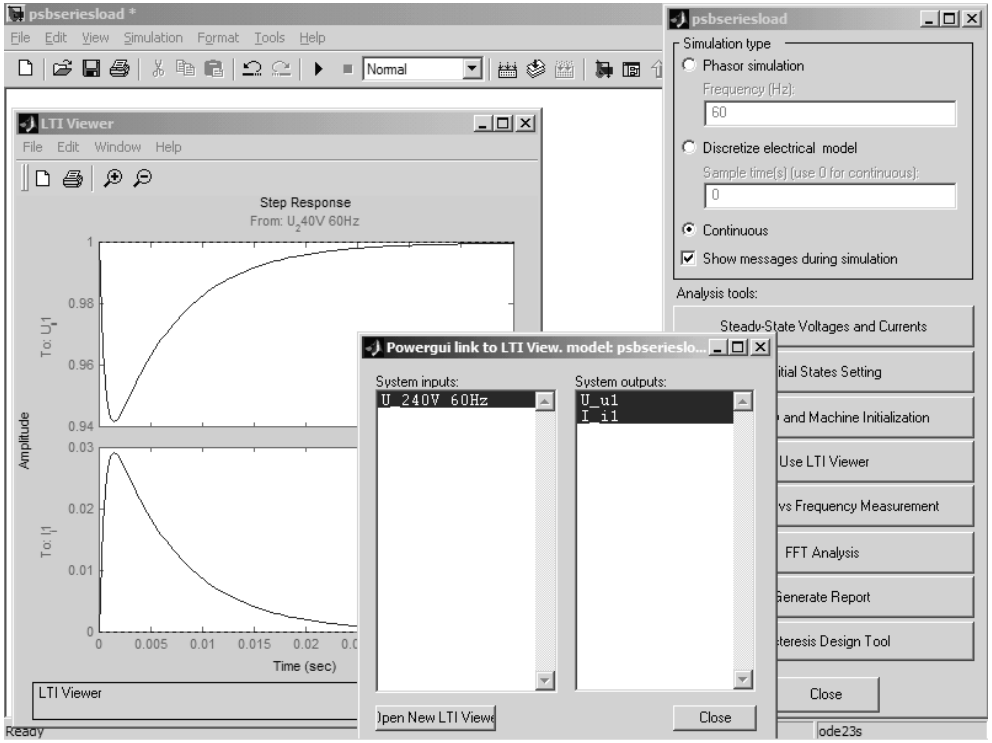


Рис. 11.13. Пример применения LTI-вьювера

При описании параметров трансформатора используется дополнительная система параметров, принятая в индустрии Запада и называемая в описании пакета pu-системой. При этом вводятся обозначения, представленные на рис. 11.17 с примерами их расчета. Запись обозначений здесь не вполне корректна, так как при формальном математическом подходе получается *pu* = *Rbase* = *Lbase*, что является грубой ошибкой. На самом деле под 1 pu в системе pu-единиц понимаются разные параметры.

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. Он соответствует теоретической модели взаимной индуктивности (рис. 11.18) и отличается от блоков линейных трансформаторов только системой параметров.

Пример моделирования цепи с блоком взаимной индуктивности дан на рис. 11.19. На нем представлены и окно параметров такого блока, и осциллограммы, иллюстрирующие работу блока при питании его от двух источников синусоидального напряжения с разными частотами и амплитудами.

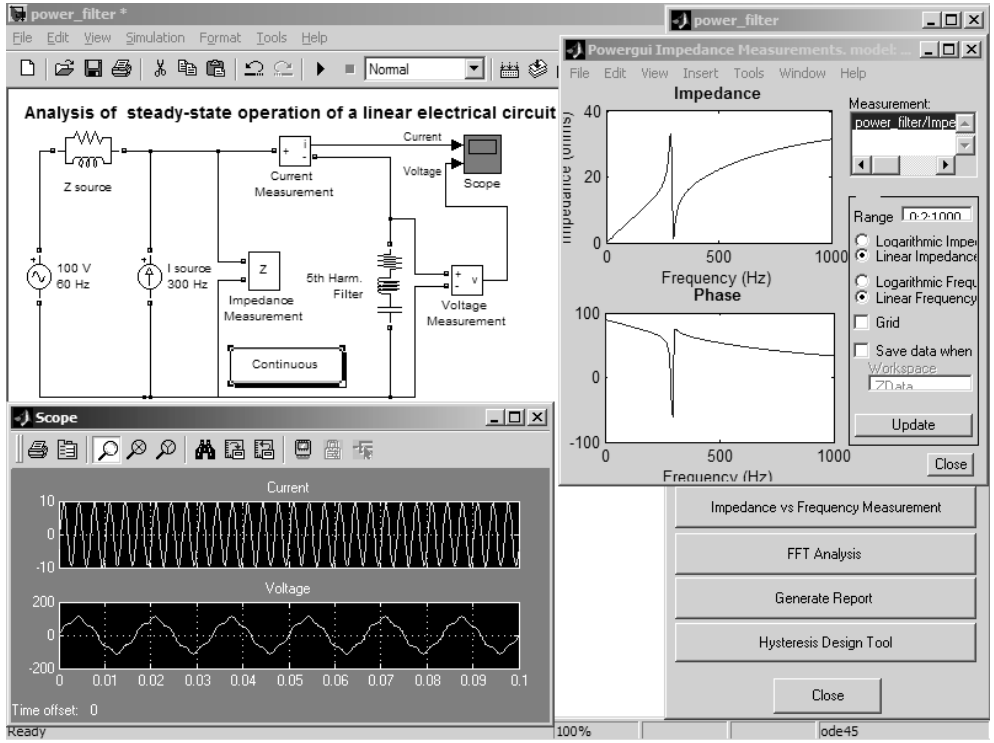


Рис. 11.14. Пример модели последовательного контура и построения характеристик его импеданса

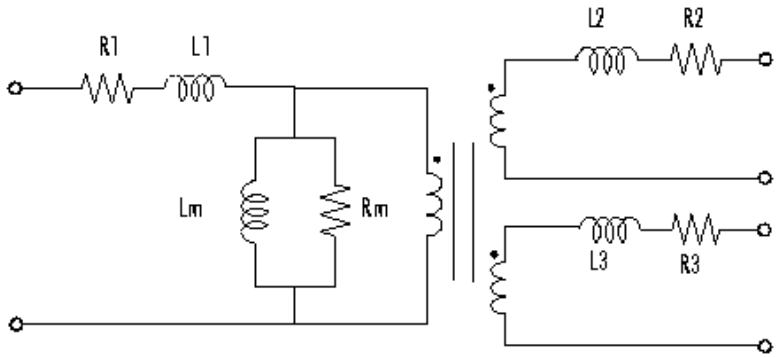


Рис. 11.15. Эквивалентная схема линейного трансформатора

Модель линейного трансформатора в значительной мере идеализирована. Модель нелинейного трансформатора **Saturable Transformer** (рис. 11.20) отличается от модели линейного трансформатора тем, что индуктивность первичной обмотки $\mathbf{L_m}$ заменена на нелинейную индуктивность $\mathbf{L_{sat}}$.

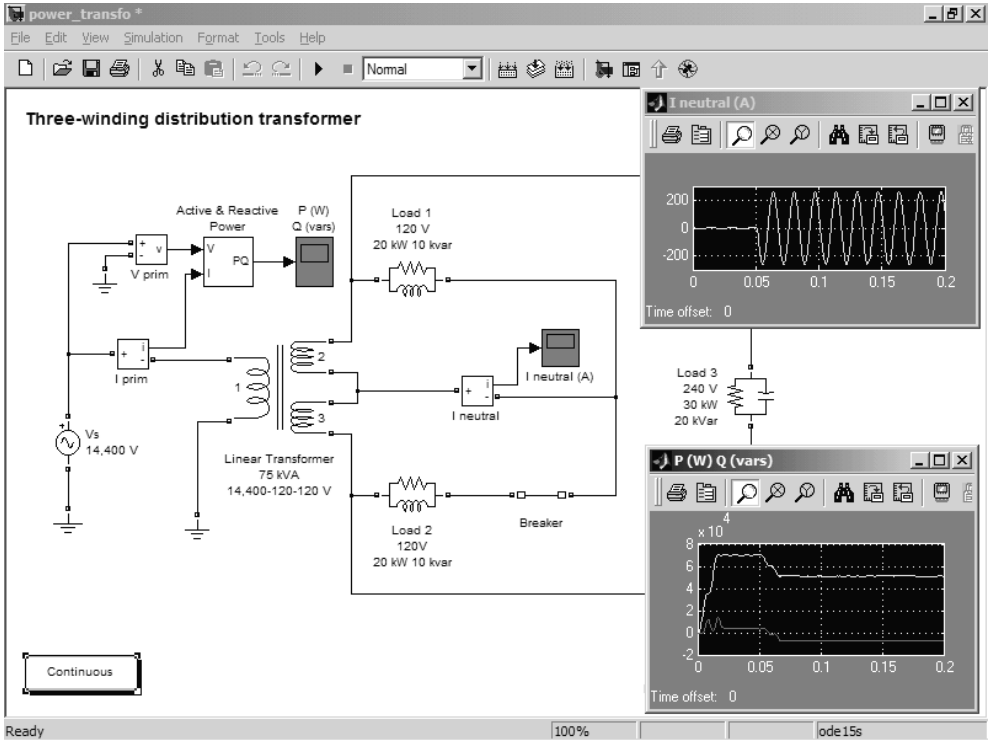


Рис. 11.16. Пример моделирования системы с линейным трансформатором

$$Rbase = 1\,pu = \frac{(Vn)^2}{Pn}$$

$$Lbase = 1\,pu = \frac{Rbase}{2\pi f_n}$$

$$Rbase = \frac{(424.35 \times 10^3)^2}{250 \times 10^6} = 720.3\Omega$$

$$Lbase = \frac{720.3}{2\pi 60} = 1.91H$$

$$R_1 = 0.002pu \times 720.3\Omega = 1.44\Omega$$

$$L_1 = 0.08pu \times 1.91H = 0.1528H$$

$$R_m = 500pu \times 720.3\Omega = 3.6 \times 10^5\Omega$$

Рис. 11.17. Формулы пересчета параметров Rbase и Lbase для трансформатора

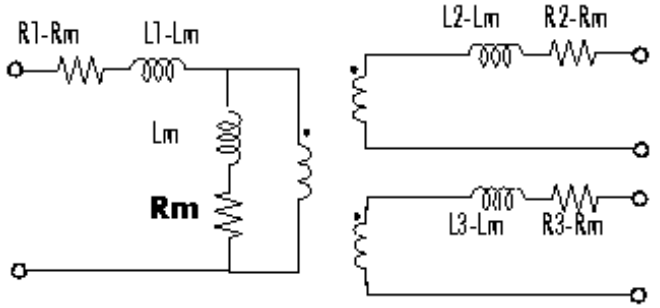


Рис. 11.18. Теоретическая модель взаимной индуктивности

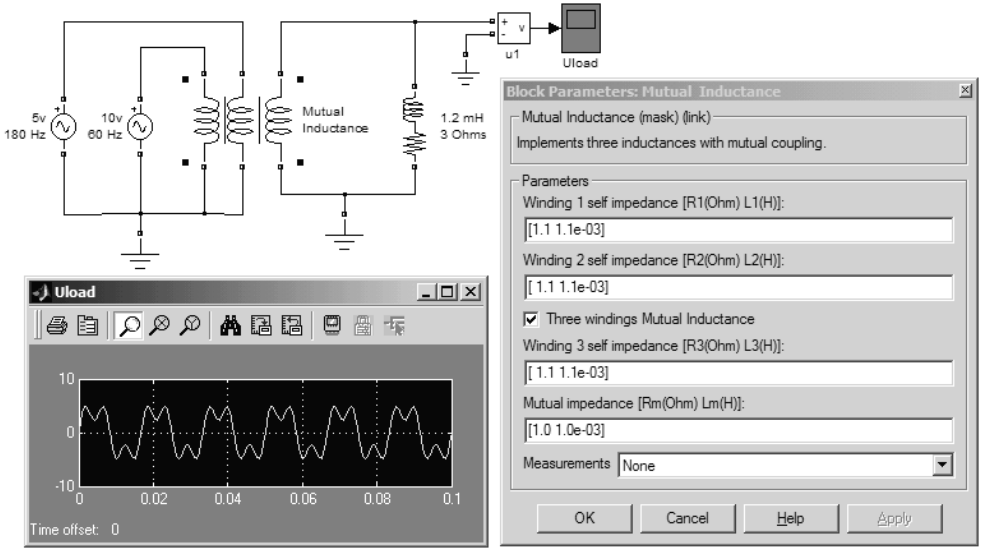


Рис. 11.19. Пример моделирования системы с блоком взаимной индуктивности

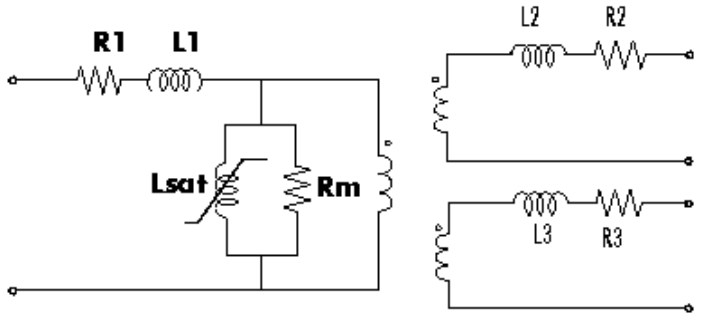


Рис. 11.20. Модель нелинейного трансформатора

Нелинейность трансформатора учитывается зависимостями, представленными на рис. 11.21. Допускаются два вида этой зависимости, отличающиеся числом опорных точек и поведением зависимости в области малых токов.

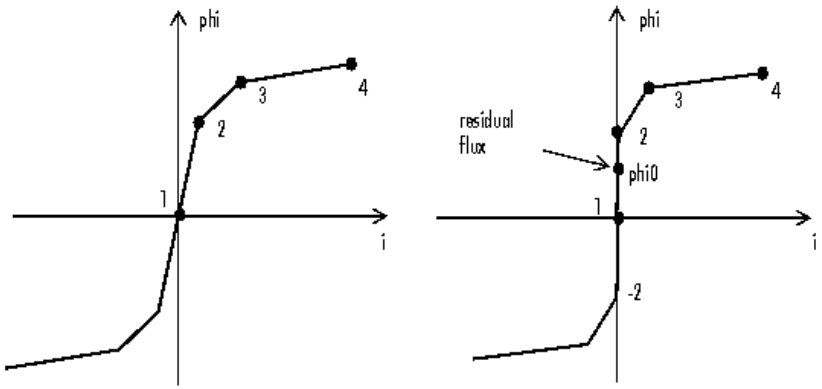


Рис. 11.21. Модели нелинейности трансформатора

На рис. 11.22 представлен пример моделирования одной фазы мощной трансформаторной подстанции с установочной мощностью 1000 МВт, построенной на основе нелинейного трансформатора. Там же показаны окно параметров нелинейного трансформатора и осциллограмма выходного напряжения. Моделируется случай работы трансформатора на холостом ходу. Обратите внимание на применение выключателя Breaker, закрывающегося после двух циклов.

Переходные процессы в этой модели могут быть получены и с помощью блока мультиметра **Multimeter** и виртуального многоканального осциллографа **Scope**. Они представлены на рис. 11.23 и хорошо отражают сложность процессов в данной модели.

К сожалению, эта модель нелинейного трансформатора не учитывает гистерезиса намагничивания сердечника и потому является лишь первым приближением к учету явлений, связанных с нелинейностью трансформатора. Рисунок 11.24 показывает модель, в которой задан гистерезис кривой намагниченности трансформатора. Модель демонстрирует переходные процессы в нелинейном трансформаторе, работающем на холостом ходу, при питании его от источника переменного напряжения через выключатель Breaker.

Построение петли гистерезиса обеспечивается с помощью блока графопостроителя `Fluc — Current characteristic`. Для организации контроля за токами и напряжениями используется блок измерения **Measurement**. Осциллограммы напряжений и токов, представленные на рис. 11.25, демонстрируют сложность переходных процессов в этой простой модели.

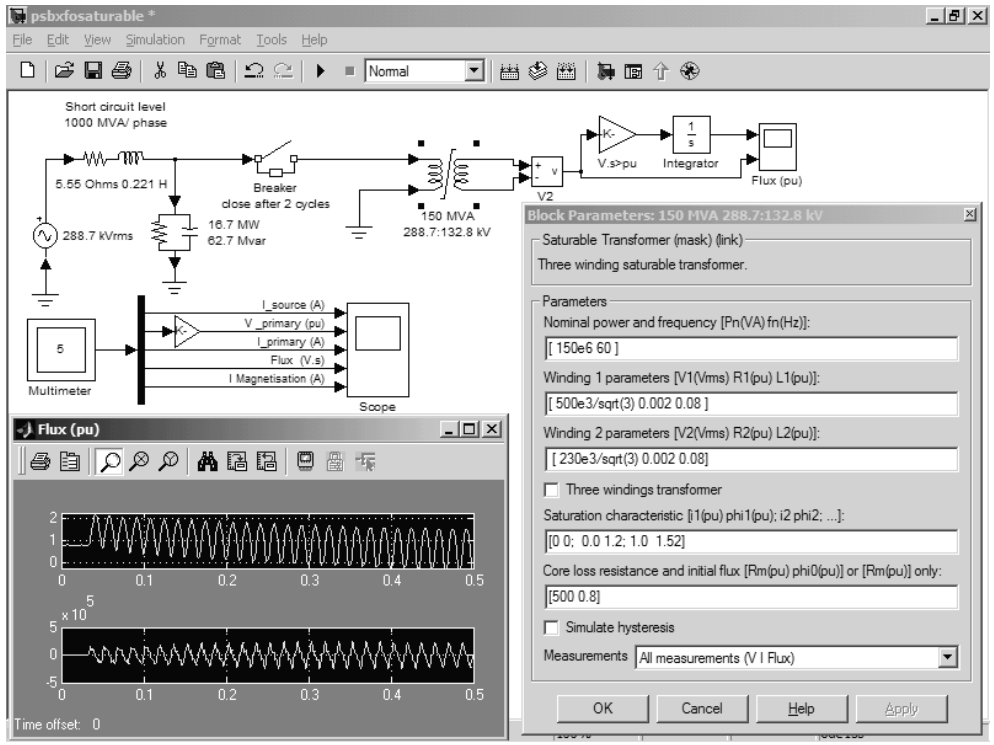


Рис. 11.22. Пример моделирования фазы трансформаторной подстанции

11.3.5. Моделирование устройств с трехфазными трансформаторами

В состав библиотеки элементов вошли также блоки линейных и нелинейных трехфазных трансформаторов. Их специфика – наличие трех первичных обмоток, на которые подаются три синусоидальных напряжения со сдвигом фаз. Пример моделирования системы с линейными трехфазными трансформаторами представлен на рис. 11.26. Модель моделирует реальную часть энергетической сети.

Предусмотрен и блок нелинейного трехфазного трансформатора. Показанная на рис. 11.27 модель системы моделирует переходные процессы в системе при включении такого трансформатора на холостой ход. Для включения используется трехфазный выключатель 3-Phase Breaker.

Осциллограммы процессов в модели рис. 11.27, показанные на рис. 11.28, свидетельствуют о сложности этих процессов, обусловленной как характером коммутационного процесса, так и нелинейностью трансформатора. В частности, это ведет к заметному искажению формы первично синусоидального сигнала.

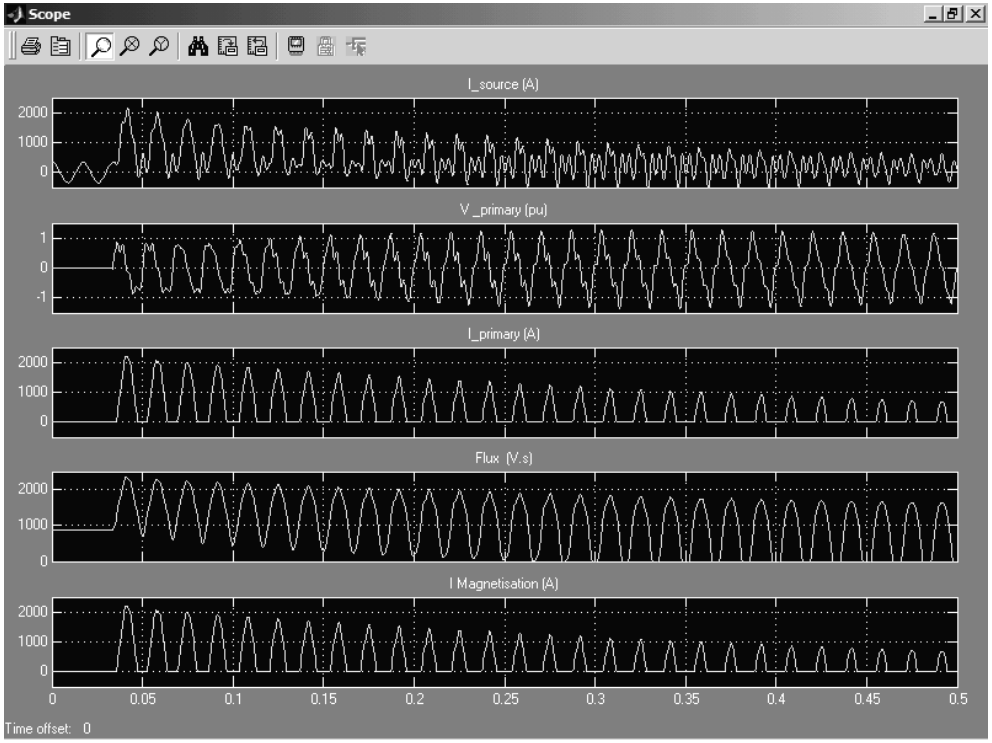


Рис. 11.23. Переходные процессы, полученные от модели рис. 11.22 с помощью мультиметра

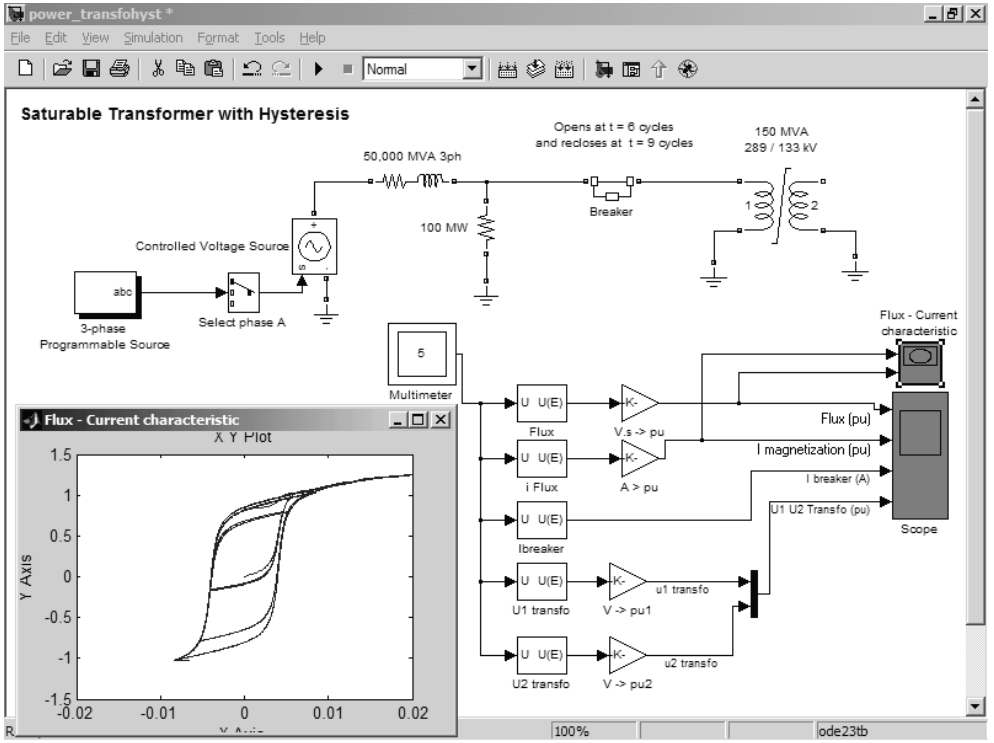


Рис. 11.24. Пример моделирования трансформатора с гистерезисом кривой намагниченности сердечника

В модель рис. 11.27 включен блок с именем **Continuous** – это названный так блок **Powergui**. С его помощью можно выполнить быстрое преобразование Фурье для выделенного фрагмента искаженного входного сигнала и получить его спектральные характеристики. Они представлены на рис. 11.29.

11.3.6. Выключатели и ограничители пиковых напряжений

Для моделирования переходных процессов в энергетических системах широко используются управляемые выключатели – блоки **Breaker** и **3-Phase Breaker**. Примеры их применения уже приводились. Рисунок 11.30 показывает простую модель с выключателем Breaker и окном установки его параметров. Действие – отключение нагрузочной RL-цепи на некоторое время – вполне очевидно.

Как видно по рис. 11.30, выключатель имеет следующие параметры:

- `Breaker resistans Ron (Ohm)` – резистивность во включенном состоянии (в омах);
- `Initial State` – начальное состояние (0 – выключено, 1 – включено);
- `Snubber resistans Rs (Ohm)` – резистивность в закрытом состоянии;
- `Snubber capacitanceans Cs (F)` – емкость, включенная последовательно с `Rs` (в F по умолчанию бесконечность – inf);
- `External control of swithing time` – опция включения внешнего контроля времени переключателя (при включенной опции появляется управляющий вход);
- `Measurement` – опция включения измерения (при ее включении появляется выход контроля).

При различных коммутациях *RLC*-цепей и трансформаторов возможно возникновение значительных выбросов напряжения и тока. В пакете SimPowerSys-

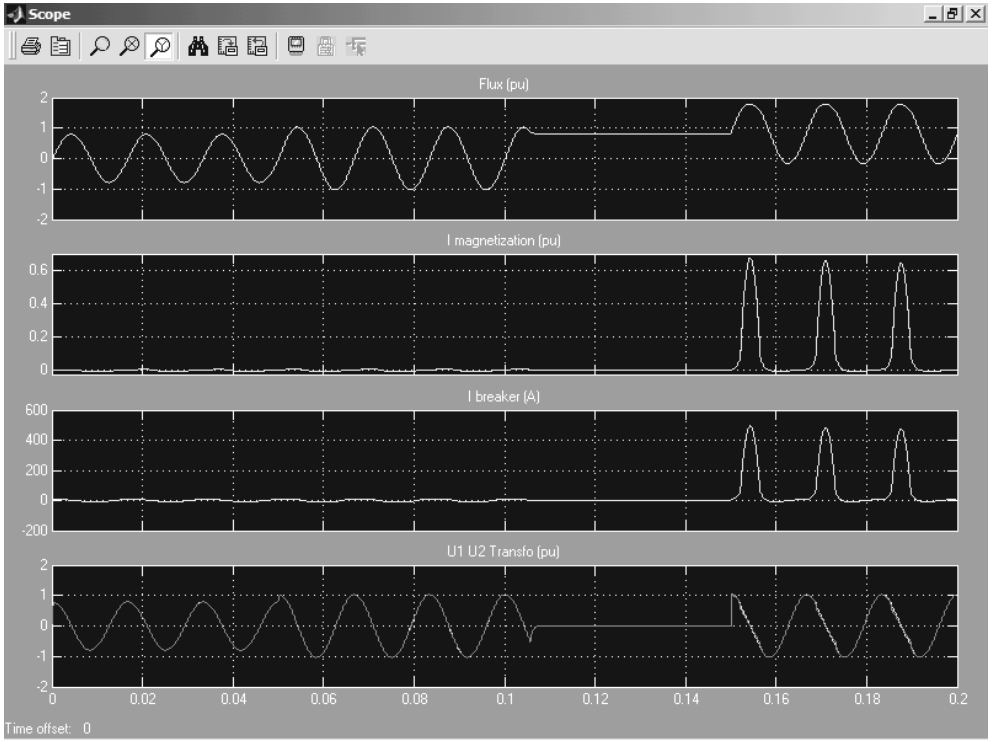


Рис. 11.25. Осциллограммы напряжений и токов модели рис. 11.51

tem можно задавать ограничители пиковых напряжений **Surge Arrester** (варисторы) с вольтамперной характеристикой, которая описывается следующим выражением:

$$\frac{V}{Vref} = K_i \left(\frac{I}{Iref} \right)^{1/\alpha_i}.$$

Такую вольтамперную характеристику можно представить в виде, показанном на рис. 11.31: слева – в обычном масштабе и справа – в логарифмическом.

На рис. 11.32 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.

Применение варисторов является одним из самых простых и эффективных методов ограничения выбросов напряжения. Варистор является типичным представителем нелинейных устройств, как и рассмотренный ранее нелинейный трансформатор. Еще один подобный блок – трехфазный замыкатель **Three-Phase**

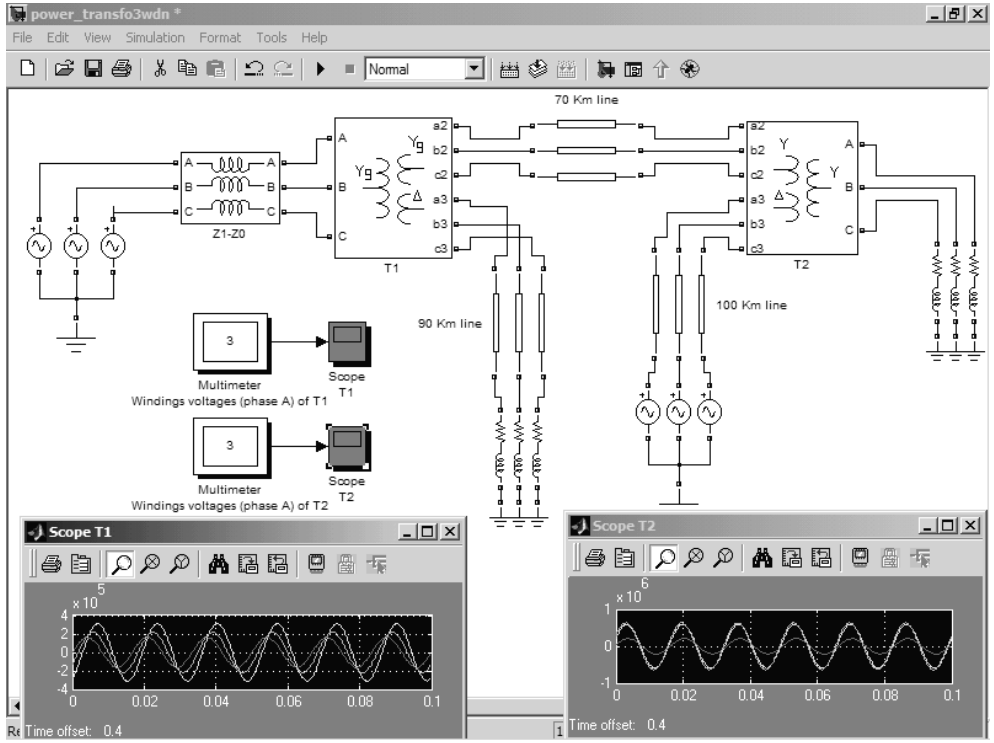


Рис. 11.26. Пример модели с линейными трехфазными трансформаторами

Fault – может использоваться для замыкания фаз поодиночке или всех сразу на землю.

11.3.7. Моделирование линий передачи

В пакете SimPowerSystem Blockset имеется возможность задания линий передачи **PI Section Line**. Эквивалентная схема такой линиии, реализованной на элементарных дискретных ячейках, представлена на рис. 11.33.

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. Линия является согласованной, когда она нагружена резистором с номиналом, равным волновому сопротивлению линии $Z_c = \sqrt{L/C}$. При согласовании на входе и на выходе линия обеспечивает минимальные искажения сигналов и отсутствие отражений.

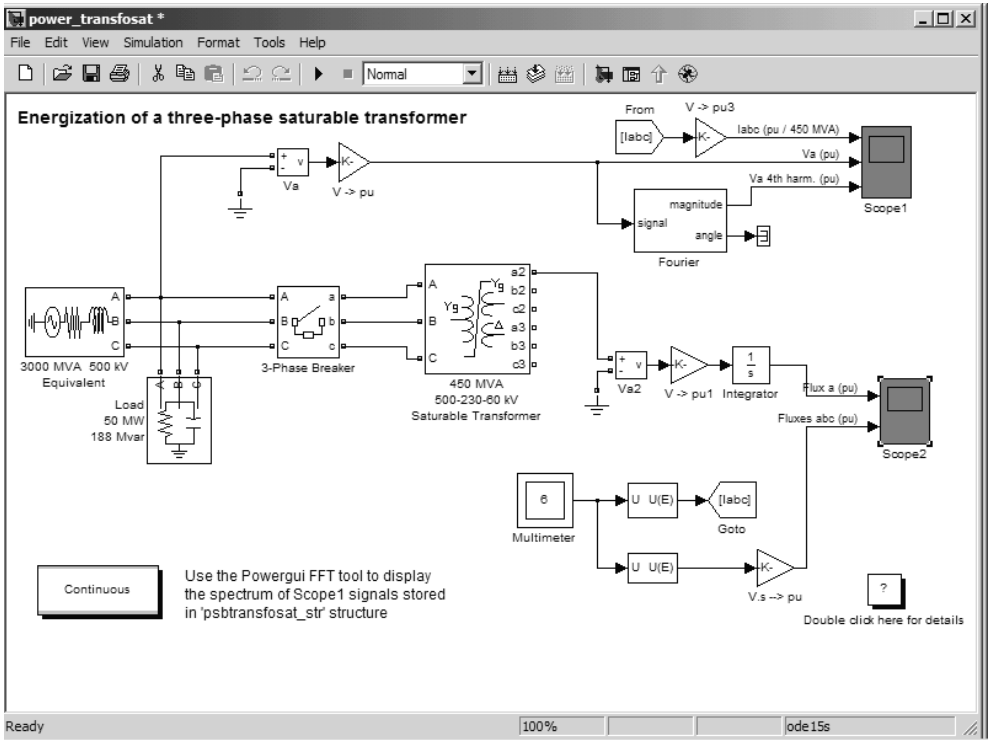


Рис. 11.27. Пример модели с нелинейным трехфазным трансформатором

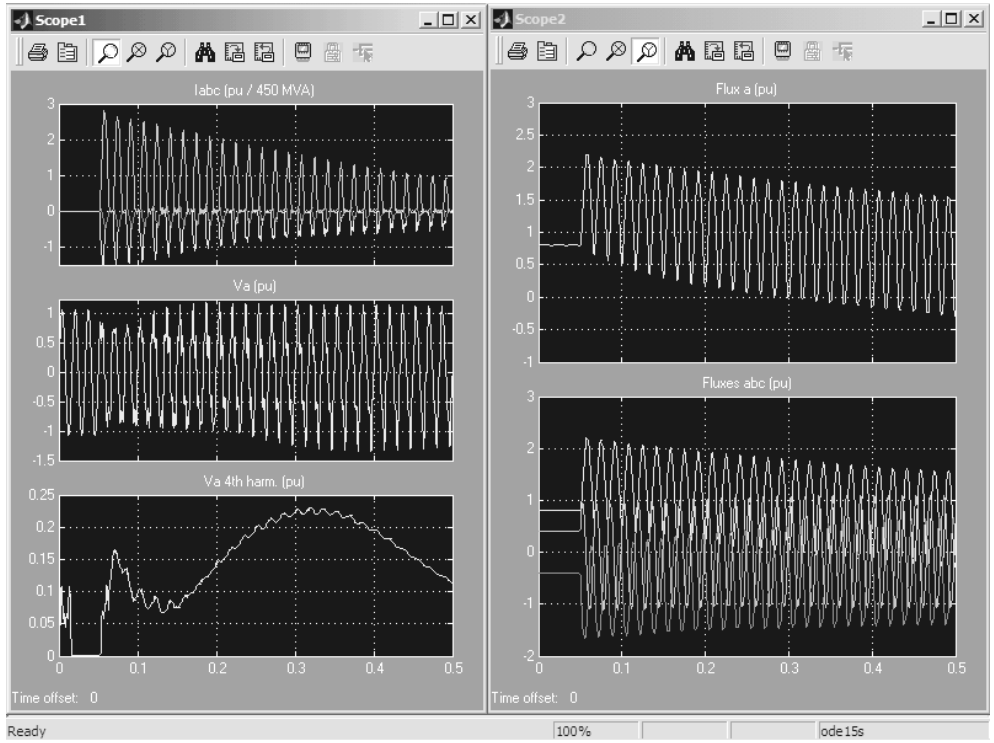


Рис. 11.28. Осциллограммы модели с нелинейным трехфазным трансформатором

Пример расчета переходных процессов при включении синусоидального напряжения на линию с сосредоточенными постоянными представлен на рис. 11.34. Там же представлены осциллограммы, иллюстрирующие работу данной модели, и окно установки ее параметров.

Максимальная частота переменного тока, который может протекать через линию данного типа, составляет $f_{\max} = N_v/8l$, 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.

Другой тип линий – линии с распределенными параметрами **Distributed Parameters Line**. Эквивалентная схема такой линии и ее математическое описание представлены на рис. 11.35.

Рисунок 11.36 дает сравнение двух типов линий одинаковой длины при однофазной передаче энергии от источника синусоидального напряжения с частотой 60 Гц и амплитудой 1 кВ. Линии имеют длину 200 км.

Осциллограммы, представленные на рис. 11.37, показывают характер переходных процессов. Сложность переходного процесса в данном примере обусловлена проявлением эффектов отражения, возникающих из-за рассогласования линии в ее начале и в конце. Для линий электропередачи согласование не используется, поскольку ведет к большим потерям энергии в цепях согласования.

На рис. 11.37 показано также окно установки параметров линий типа **Distributed Parameters Line**. Оно имеет дополнительный параметр – число фаз линии.

11.3.8. Моделирование линии передачи с компенсаторами

На рис. 11.38 показан пример моделирования переходных процессов в высоковольтной линии передачи электроэнергии с напряжением 735 кВ, имеющей два участка длиной 150 км каждый и индуктивные компенсаторы. Между ними расположен емкостный компенсатор, оформленный в виде подсистемы – ее модель показана в левом нижнем углу.

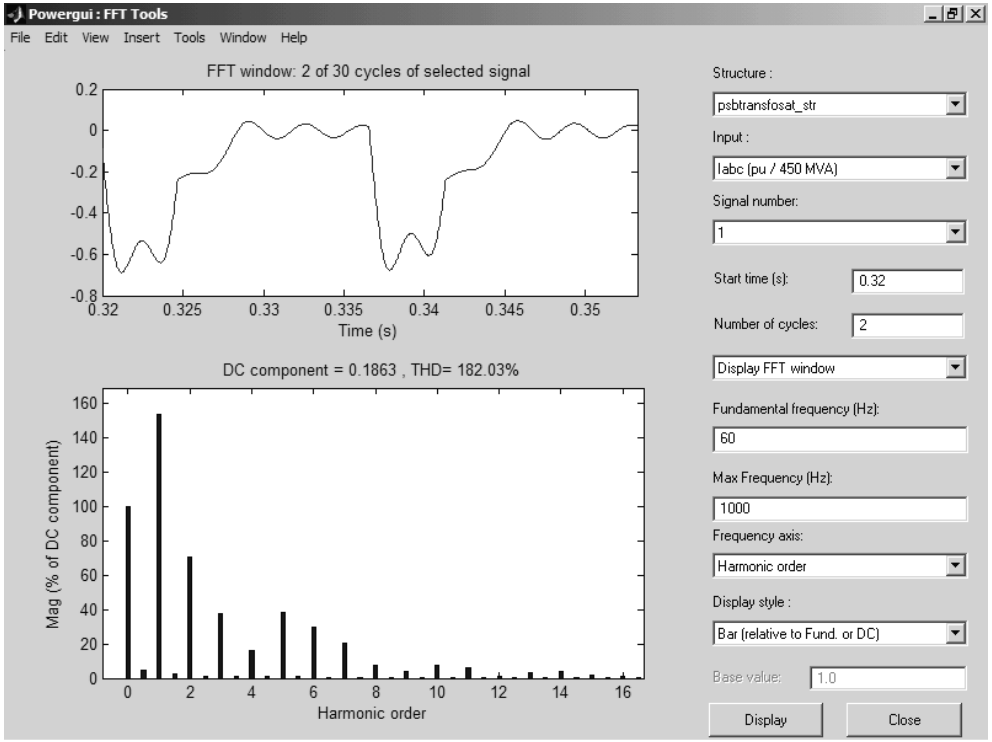


Рис. 11.29. Спектральные характеристики для фрагмента входного сигнала модели рис. 11.27

Осциллограммы, иллюстрирующие работу линии с компенсаторами, представлены на рис. 11.39. Они позволяют судить о сложности переходных процессов, возникающих при коммутации линий электропередачи.

Обратите внимание на блок **Impedance Measurement**. Это один из блоков библиотеки **Measurement**. Блок позволяет измерять импеданс подключенной к нему цепи на разных частотах и с помощью блока **Powergui** строить зависимости модуля импеданса и фазы от частоты – рис. 11.40.

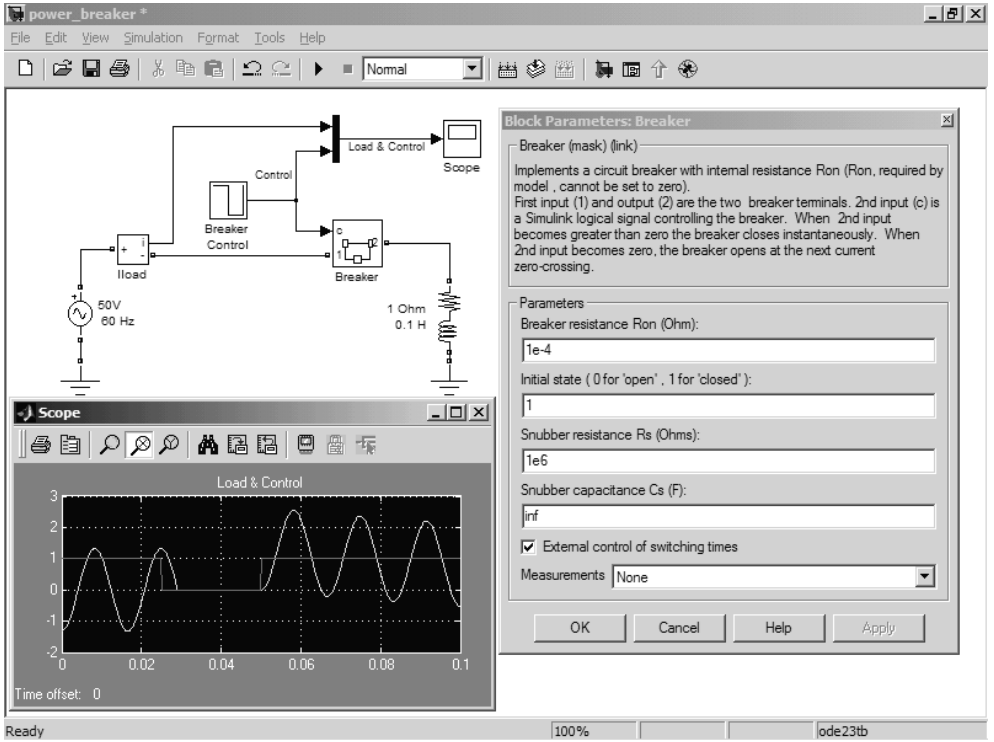


*Рис. 11.30. Простая модель с выключателем **Breaker***

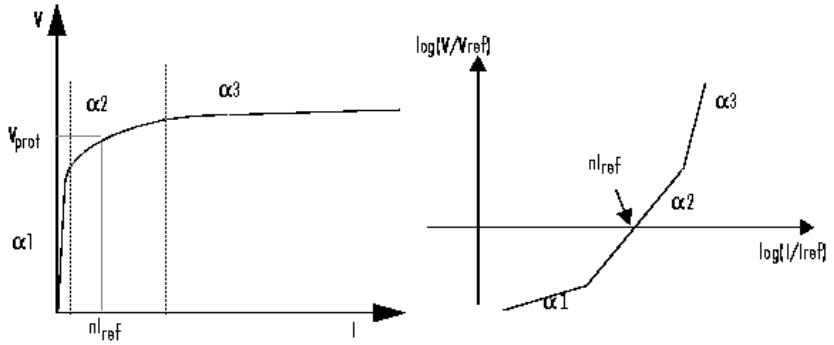


Рис. 11.31. Вольтамперные характеристики нелинейного ограничителя в линейном (слева) и в логарифмическом (справа) масштабах

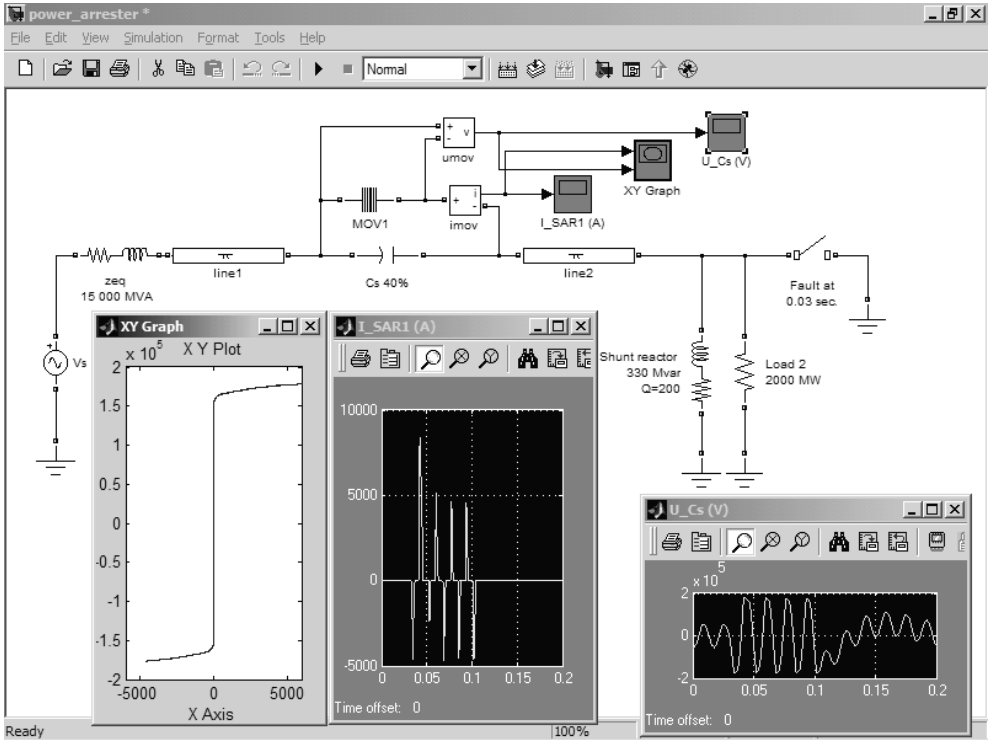


Рис. 11.32. Моделирование электрической системы с варистором – ограничителем выбросов напряжения

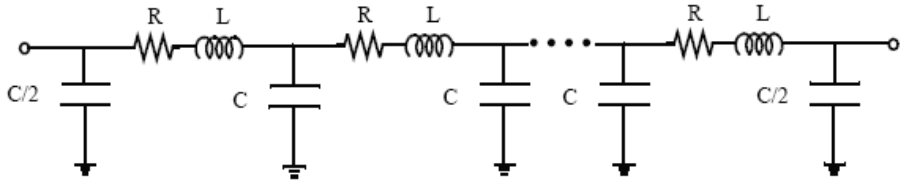


Рис. 11.33. Эквивалентная схема линии с сосредоточенными параметрами

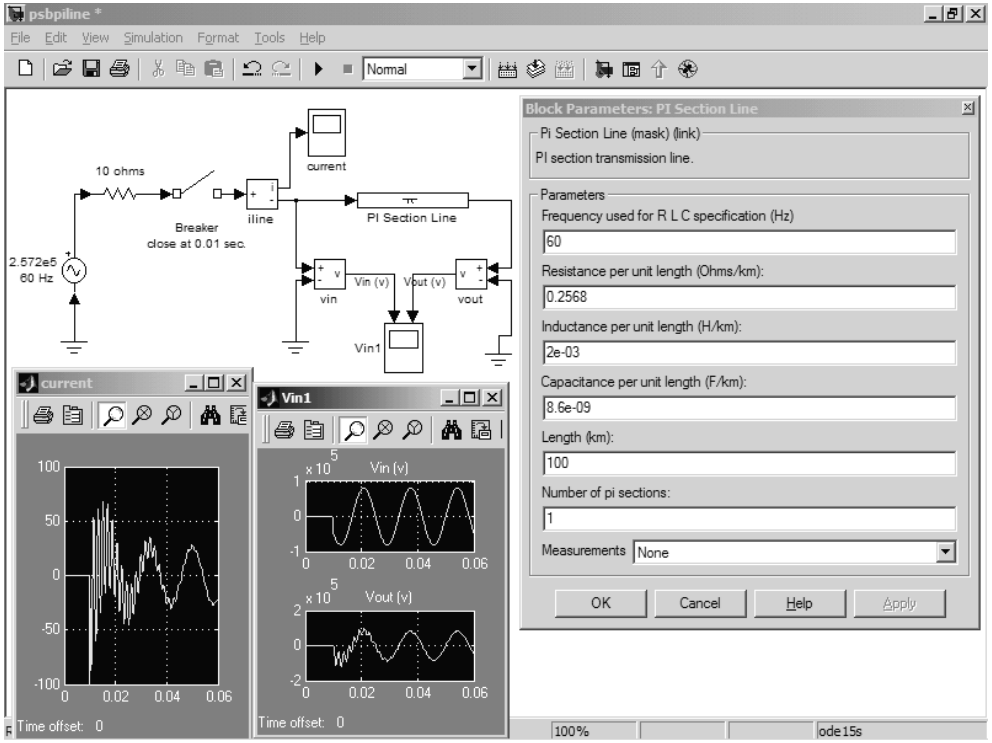
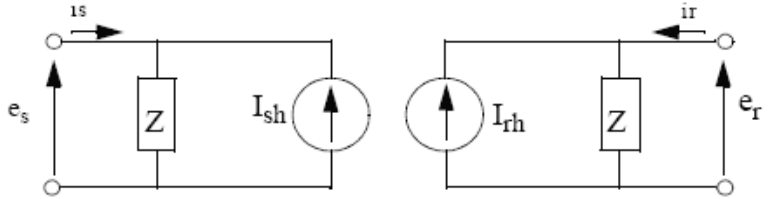


Рис. 11.34. Моделирование переходного процесса при включении синусоидального напряжения на линию с сосредоточенными параметрами



$$I_{sh}(t) = \left(\frac{1+h}{2}\right)\left[\frac{1}{Z}e_r(t-\tau) + hi_r(t-\tau)\right] + \left(\frac{1-h}{2}\right)\left[\frac{1}{Z}e_s(t-\tau) + hi_s(t-\tau)\right]$$

$$I_{rh}(t) = \left(\frac{1+h}{2}\right)\left[\frac{1}{Z}e_s(t-\tau) + hi_s(t-\tau)\right] + \left(\frac{1-h}{2}\right)\left[\frac{1}{Z}e_r(t-\tau) + hi_r(t-\tau)\right]$$

где $Z = Z_C + \frac{R}{4}$ $\quad h = \frac{Z_C - \frac{R}{4}}{Z_C + \frac{R}{4}}$ $\quad Z_C = \sqrt{\frac{L}{C}}$ $\quad \tau = d\sqrt{LC}$

Рис. 11.35. Эквивалентная схема линии с сосредоточенными параметрами

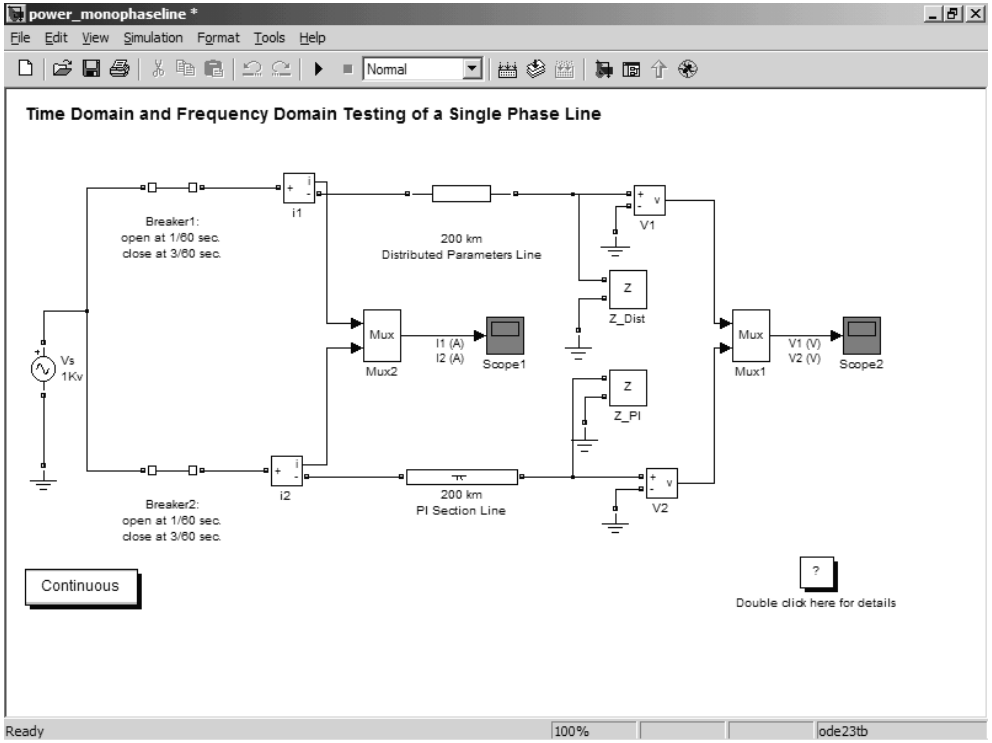


Рис. 11.36. Пример использования линии с распределенными параметрами

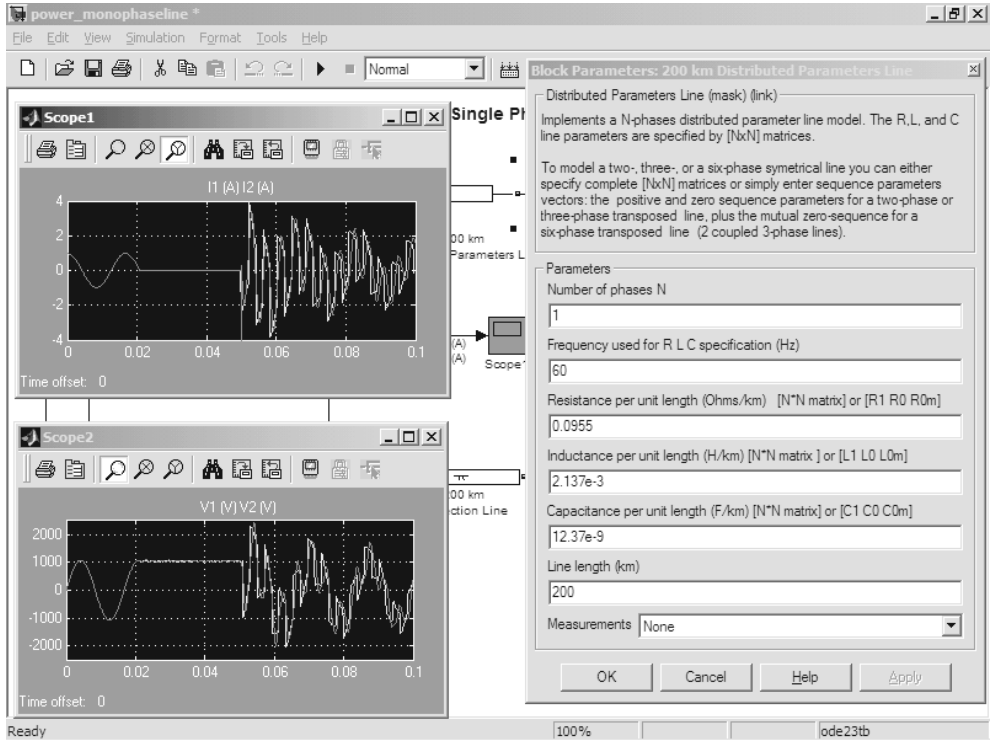


*Рис. 11.37. Осциллограммы модели рис. 11.36 и окно установки параметров линии **Distributed Parameters Line***

11.4. Моделирование систем и устройств энергетической электроники

11.4.1. Состав библиотеки энергетической электроники

Современная силовая электроника основана на импульсном способе преобразования электрической энергии, обеспечивающем высокий коэффициент полезного действия преобразовательных устройств (теоретически до 100%). Поэтому в библиотеку рассматриваемого пакета включен достаточно представительный набор блоков коммутирующих устройств. Активизация пиктограммы **Power Electronics** открывает окно с пиктограммами моделей управляемых ключей (рис. 11.41).

В библиотеке **Power Electronics** 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;
- **Three-Level Bridge** – модуль трехфазного моста.

Внизу окна этой библиотеки имеются два раздела библиотеки, представляющие внешние маскированные блоки – маски:

- `Discrete Control blocks` – блоки контроля дискретные (25 блоков);
- `Control blocks` – блоки контроля непрерывные (14 блоков).

Для контроля над работой устройств имеется специальный раздел библиотеки `Measurements` **(Измерения)**. Окно этого раздела представлено на рис. 11.41 справа и содержит следующие блоки:

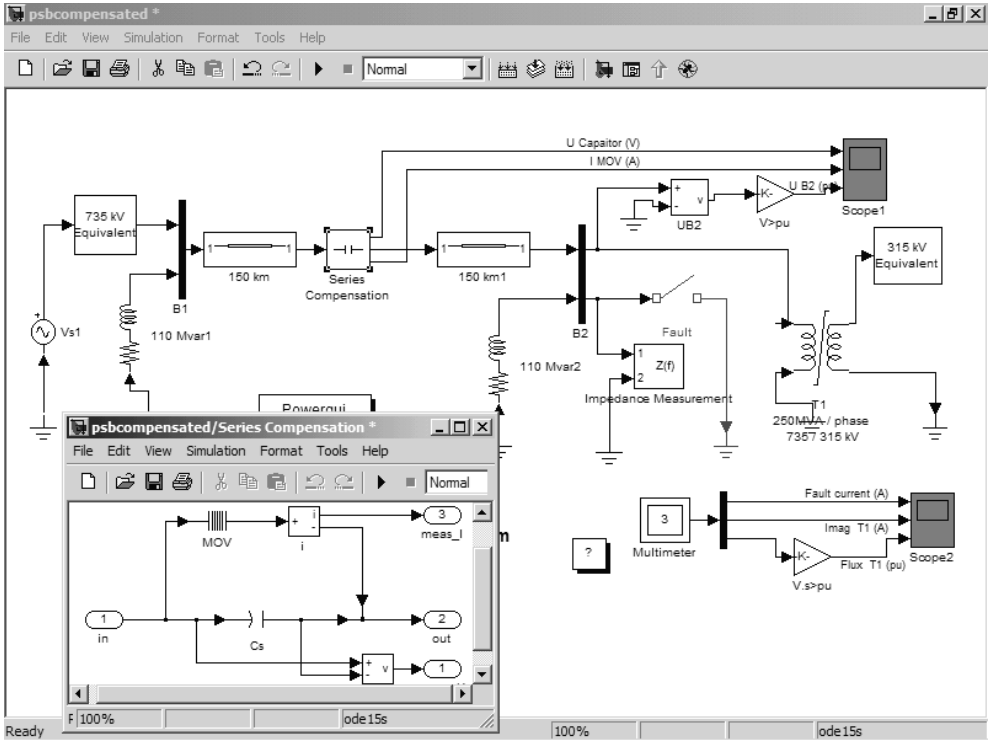


Рис. 11.38. Моделирование линии электропередачи с компенсаторами

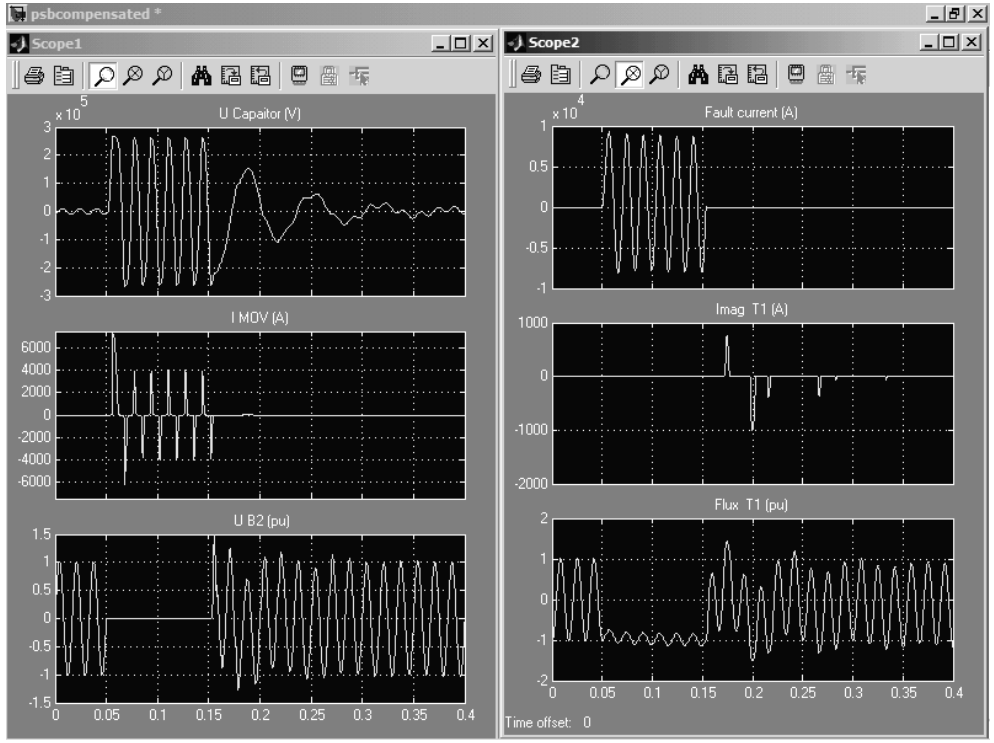


Рис. 11.39. Осциллограммы модели рис. 11.38

- `Current Measurement` – блок измерения тока;
- `Voltage Measurement` – блок измерения напряжения;
- `Impedance Measurement` – блок измерения импеданса;
- `Multimeters` – мультиметр;
- `Theree-Phase V-I Measurement` – измеритель напряжений/токов трехфазный.

Применение этих блоков вполне очевидно и уже не раз демонстрировалось. Ряд блоков дополнительной библиотеки powerlib_extras будет более детально описан в конце данного урока.

11.4.2. Моделирование простых ключевых устройств

Все модели коммутирующих элементов содержат формирующую траекторию рабочей точки и нередко гасящую выбросы напряжения последовательную RsCs-цепь, которая подключается к силовым выводам моделей. Задание бесконечного значения $\mathbf{C_s}$ и нулевого $\mathbf{R_s}$ закорачивает модель (пиктограмма устройства при этом заменяется пиктограммой проводника). Задание $\mathbf{C_s}$ нулю исключает эту цепь из анализа. 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.

Идеальный ключ **Ideal Swith** моделирует ключ, который во включенном состоянии имеет сопротивление $\mathbf{R_{on}}$ и индуктивность $\mathbf{L_{on}}$. Сопротивление $\mathbf{R_{on}}$ позволяет приближенно учитывать статические потери в ключе во включенном состоянии, а индуктивность $\mathbf{L_{on}}$ – инерционные процессы при переключении. Задание $\mathbf{L_{on}}$=0 недопустимо, поскольку ведет к неразрешимости системы уравнений, описывающих работу электрических цепей, из-за деления на нуль. Сопротивление ключа в выключенном состоянии считается равным бесконечности. Если требуется задать конечное сопротивление, то это легко моделируется включением резистора соответствующего номинала параллельно зажимам ключа. Рисунок 11.42 дает пример моделирования цепи с идеальным ключом.

В исходном состоянии ключ может быть закрыт или открыт в зависимости от параметра **Initial State**. Кроме того, можно использовать последовательную RsCs-

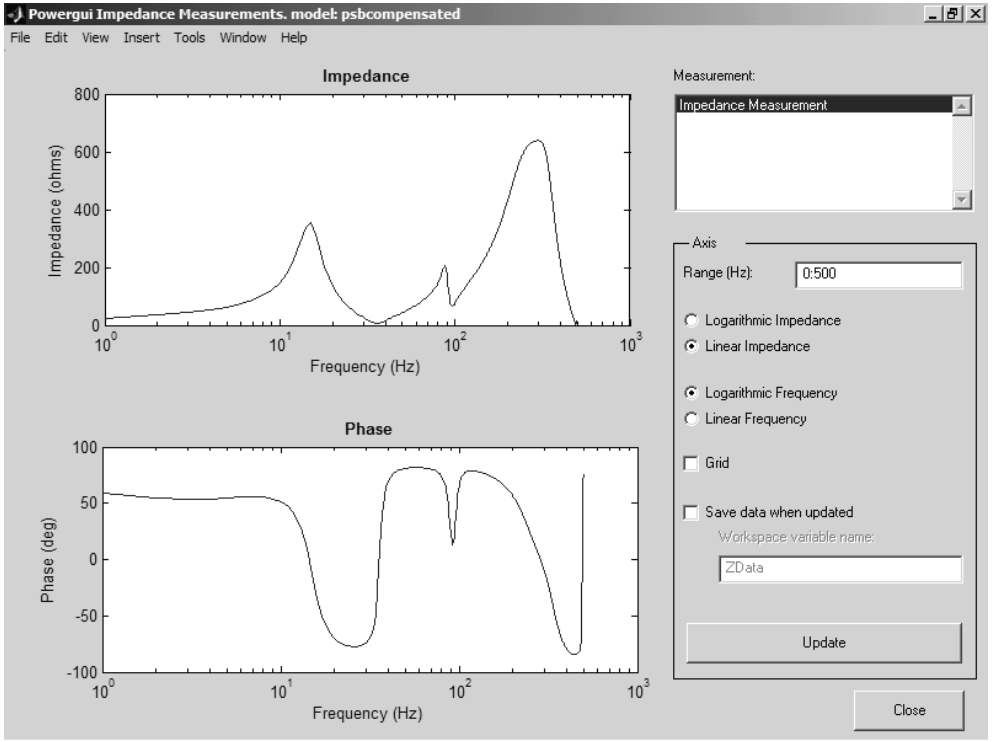


Рис. 11.40. Зависимости модуля и фазы импеданса от частоты

цепь, которая включается параллельно ключу для ограничения выбросов напряжения и подавления дуги.

Модель диода **Diode** представляет собой последовательно соединенные источник напряжения $\mathbf{V_f}$ на диоде в прямом включении, резистор $\mathbf{R_{on}}$ и паразитную индуктивность $\mathbf{L_{on}}$ в прямом направлении, когда диод проводит ток. В обратном направлении сопротивление диода считается бесконечно большим. Предусмотрен учет включения параллельно диоду корректирующей последовательной цепи RsCs (эта цепь является внутренней и в составе модели не показана). Следует отметить, что модель диода довольно приближенная. Вполне очевидная модель с диодом представлена на рис. 11.43. Там же показано окно установки параметров диода и осциллограммы, иллюстрирующие работу модели.

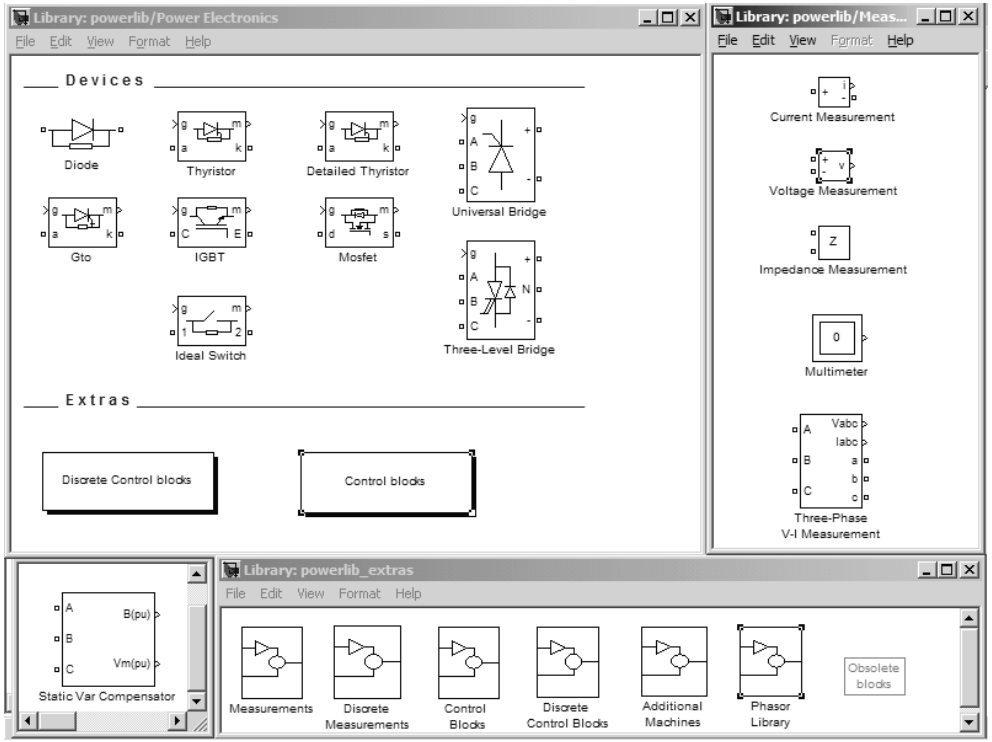


Рис. 11.41. Состав основных библиотек блоков энергетической электроники

11.4.3. Моделирование устройств с мощными ключевыми полевыми транзисторами

Полевые транзисторы с изолированным затвором в последнее время стали основным типом переключающих транзисторов малой и средней (а иногда и большой) мощности. В пакете SimPowerSystem Blockset предусмотрена простая модель полевого транзистора **Mosfet**. Фактически он рассматривается как силовой ключ с сопротивлением $\mathbf{R_{on}}$ и индуктивностью $\mathbf{L_{on}}$ во включенном состоянии и бесконечно большим сопротивлением в выключенном состоянии. Можно также задать включение параллельно транзистору диода, открытого при закрытом транзисторе и характеризующегося сопротивлением $\mathbf{R_d}$. Наконец, можно добавить подключенную к выводам сток–исток корректирующую последовательную RsCs-цепь.

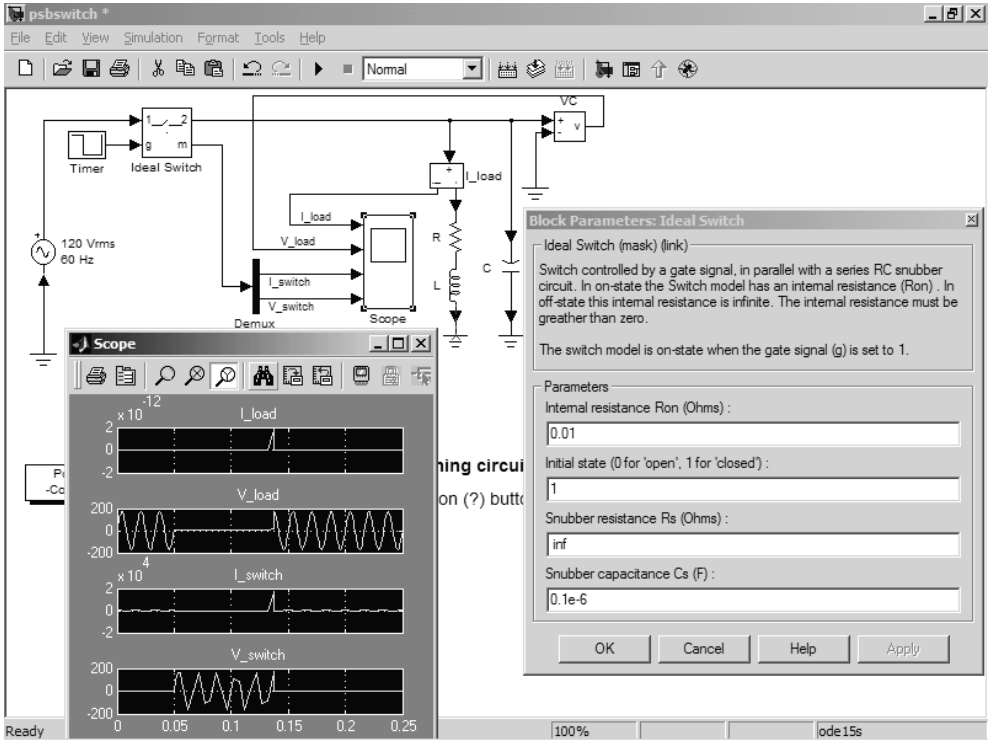


Рис. 11.42. Моделирование цепи с идеальным ключом

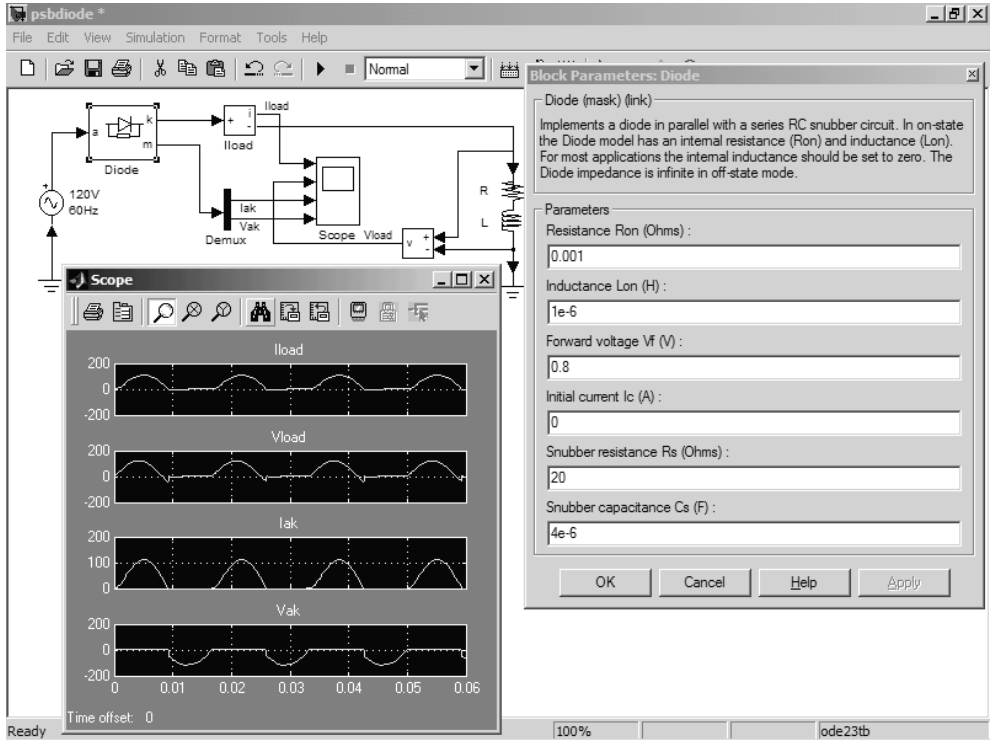


Рис. 11.43. Моделирование цепи с диодом

Пример модели с полевым транзистором представлен на рис. 11.44. Моделируется упрощенная схема квазирезонансного инвертора с переключением по нулю тока. На рисунке даны также фазовая траектория и осциллограммы процессов в инверторе.

Рисунок 11.45 показывает окно установки параметров полевого транзистора. Блок полевого транзистора маскированный, и на рис. 11.45 представлена его модель. Она содержит обращение к внутренней модели.

11.4.4. Моделирование устройств с тиристорами

Модель тиристора **Thyristor** также построена на основе идеального ключа с элементами, имитирующими остаточные параметры включенного тиристора, – рис. 11.46. Это сопротивление во включенном состоянии $\mathbf{R_{on}}$, индуктивность $\mathbf{L_{on}}$ и падение напряжения в прямом направлении $\mathbf{V_f}$. 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 $\mathbf{I_c}$, с которого начинается моделирование (по умолчанию 0, то есть моделирование начинается при закрытом тиристоре).

Имеется и более детальная модель тиристора – **Detail Thyristor**. Окно параметров этой модели имеет два дополнительных параметра: ток $\mathbf{I_l}$, при котором тиристор выключается, и время выключения $\mathbf{T_q}$ (в секундах), которое характеризует задержку выключения. Эта модель позволяет более точно моделировать переходные процессы в схемах с тиристорами, что важно при моделировании устройств, работающих с повышенными частотами переключения – сотни герц и выше. Для моделирования устройств, работающих на частоте промышленной сети (50 или 60 Гц), вполне удовлетворительные результаты дает упрощенная модель тиристора.

Запираемые тиристоры **Gto** – сравнительно новый и перспективный тип мощных силовых коммутирующих элементов. Они имеют управляющие сигналы малой мощности и способны переключать бóльшие напряжения и токи, чем полевые транзисторы с изолированным затвором. Однако для этих устройств характерно

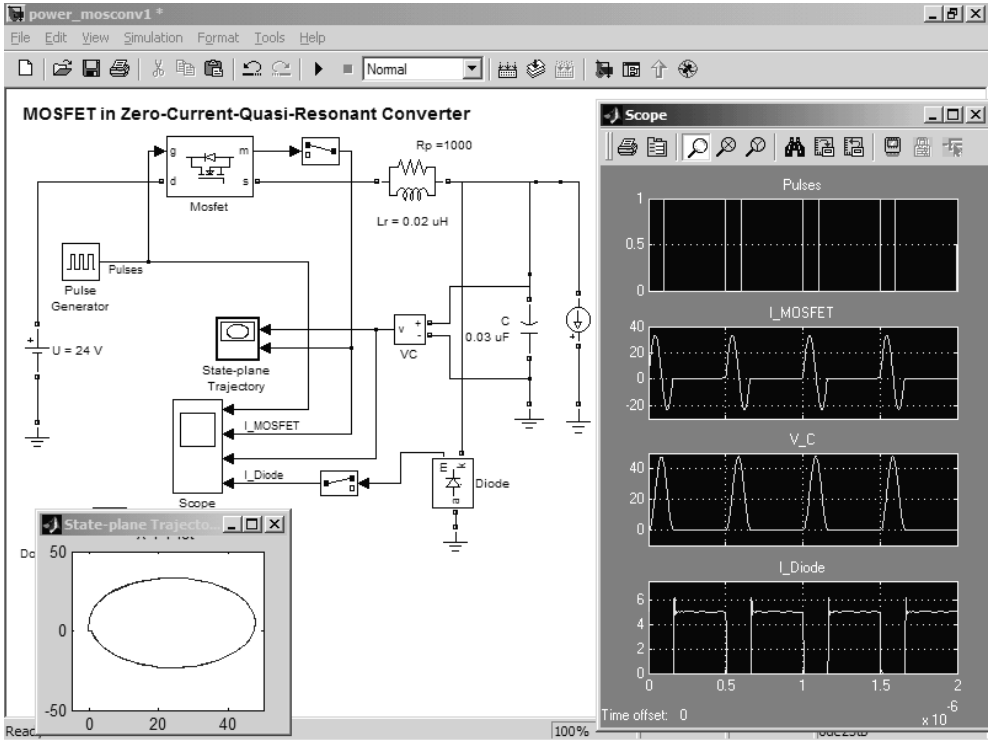


Рис. 11.44. Моделирование квазирезонансного инвертора на полевом транзисторе

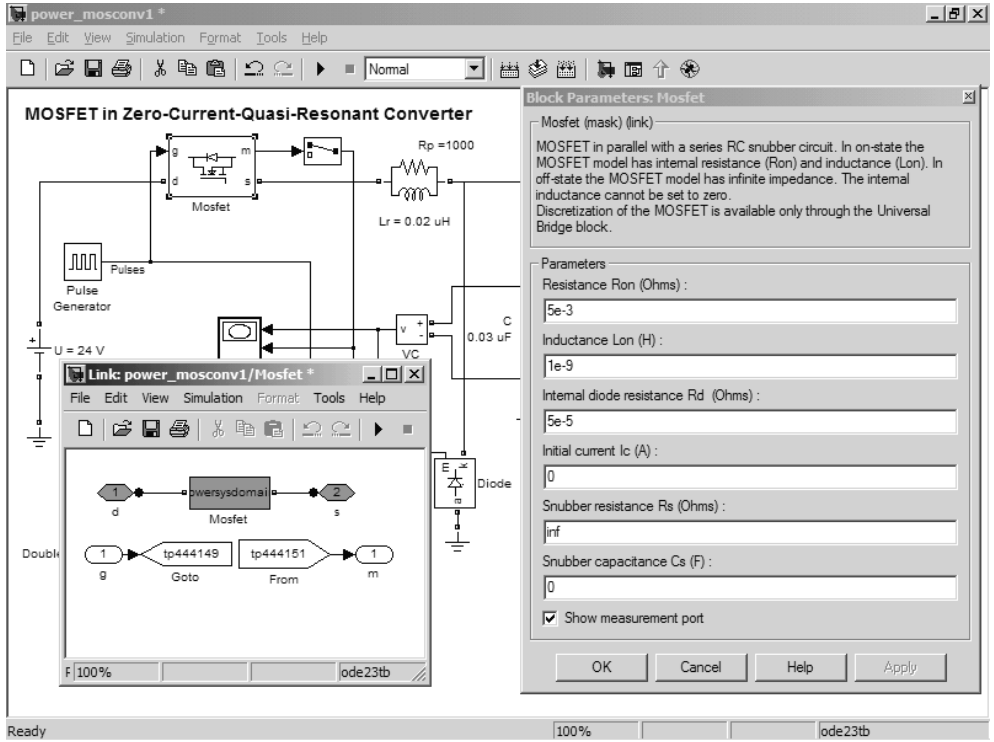


Рис. 11.45. Окно параметров и модель мощного полевого транзистора

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 ($\mathbf{T_f}$) и время окончательного спада тока до нуля ($\mathbf{T_t}$). Временная зависимость спада тока приближенно описывается двумя линейными участками с указанными длительностями.

11.4.5. Моделирование устройств с запираемыми Gto модулями

На рис. 11.47 показана схема преобразователя постоянного напряжения, в котором используется **Gto**-модуль. Осциллограммы поясняют работу этой схемы. Нагрузкой преобразователя являются 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.

11.4.6. Моделирование устройств с силовыми IGBT-модулями

Силовой модуль **IGBT** (рис. 11.48) – новый перспективный элемент энергетической электроники. Он создан на основе комбинации биполярных транзисторов с полевыми. В прежних версиях описываемого пакета блок **IGBT**-модуля отсутствовал.

Статическая вольтамперная характеристика модуля может быть представлена в виде двух отрезков прямых. Горизонтальный участок характерен для выклю-

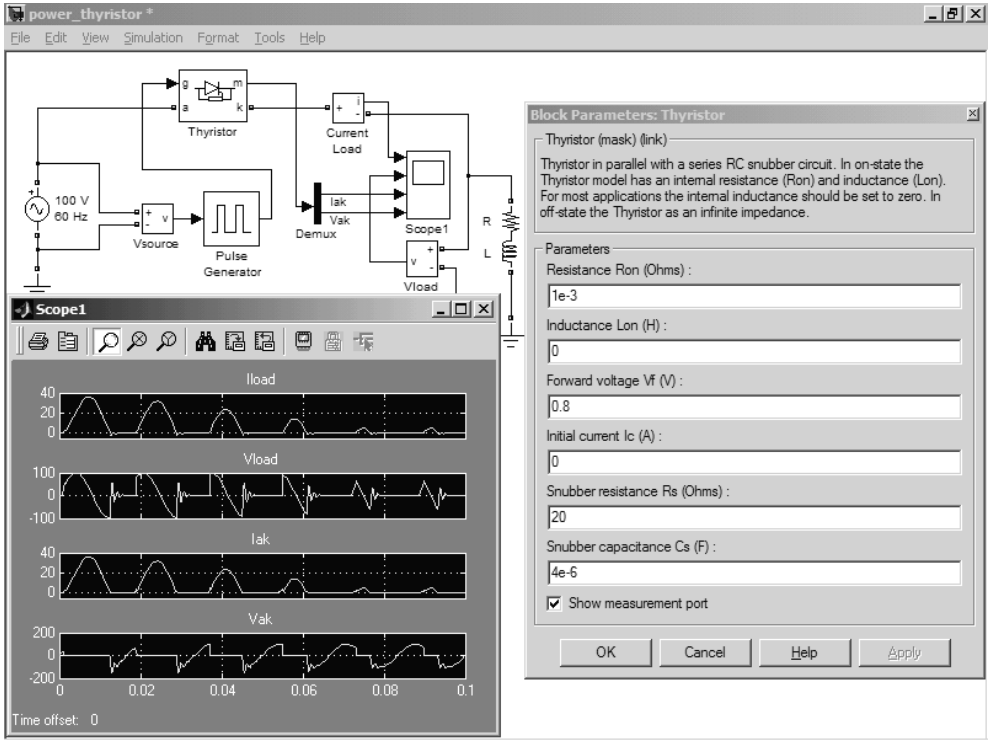


Рис. 11.46. Модель тиристорного регулятора и окно параметров тиристора

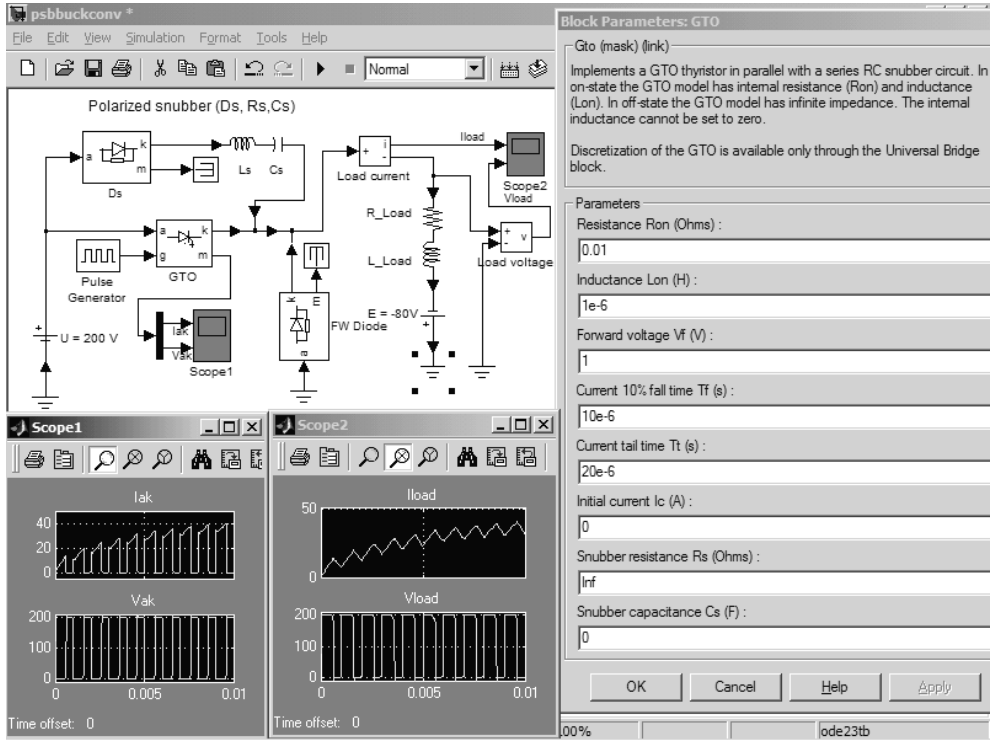


Рис. 11.47. Моделирование преобразователя постоянного напряжения с Gto-модулем

ченного 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 $\mathbf{V_f}$.

Для устройств **IGBT**, как и для **GTO**, характерен довольно медленный процесс выключения, состоящий из двух стадий. Его временные диаграммы, представленные на рис. 11.49, демонстрируют два важных временных параметра блока **JGBT** – время спада тока $\mathbf{T_f}$ и $\mathbf{T_t}$.

На рис. 11.50 представлен пример моделирования повышающего напряжение импульсного преобразователя на основе **IGBT**-модуля. Это типичный обратноходовой преобразователь с индуктивным накопителем энергии.

Приведенные на рис. 11.50 осциллограммы показывают наличие довольно длительного переходного процесса, в ходе которого наблюдается выброс выходного напряжения, почти вдвое превышающий установившееся значение.

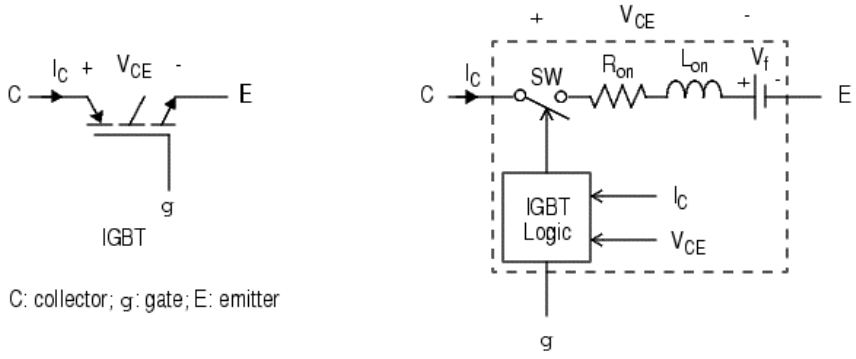


*Рис. 11.48. Блок **IGBT**: обозначение (слева) и модель (справа)*

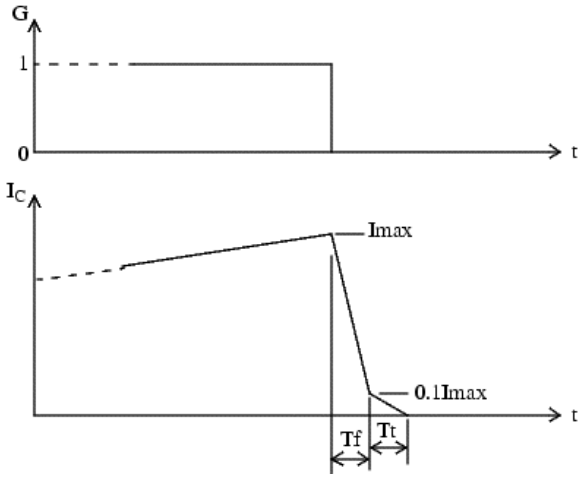


*Рис. 11.49. Временные диаграммы выключения модуля **IGBT***

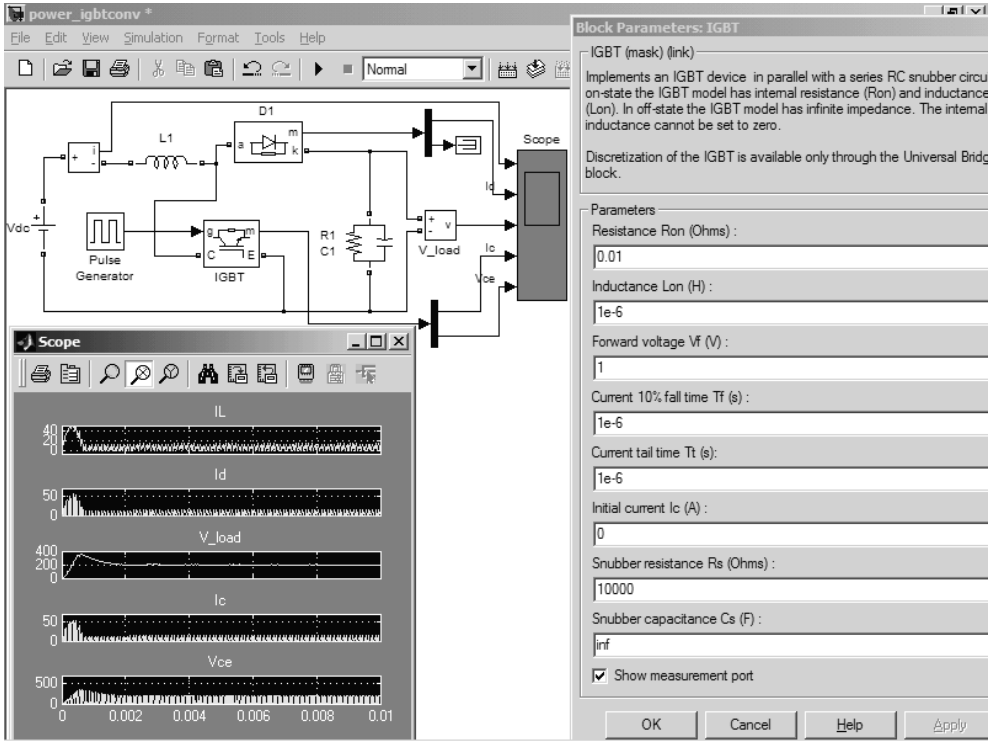


*Рис. 11.50. Повышающий напряжение импульсный преобразователь на основе **IGBT**-модуля*

11.4.7. Моделирование устройств с мостовыми модулями

Учитывая широкое применение мостовых схем в энергетической электронике, в новую версию пакета Power System Blockset был включен универсальный мостовой модуль **Universal Bridge**. Окно установки параметров этого блока показано на рис. 11.51. На нем открыт список **Power Electronics Device**, который задает выбор одного из описанных выше ключевых приборов.

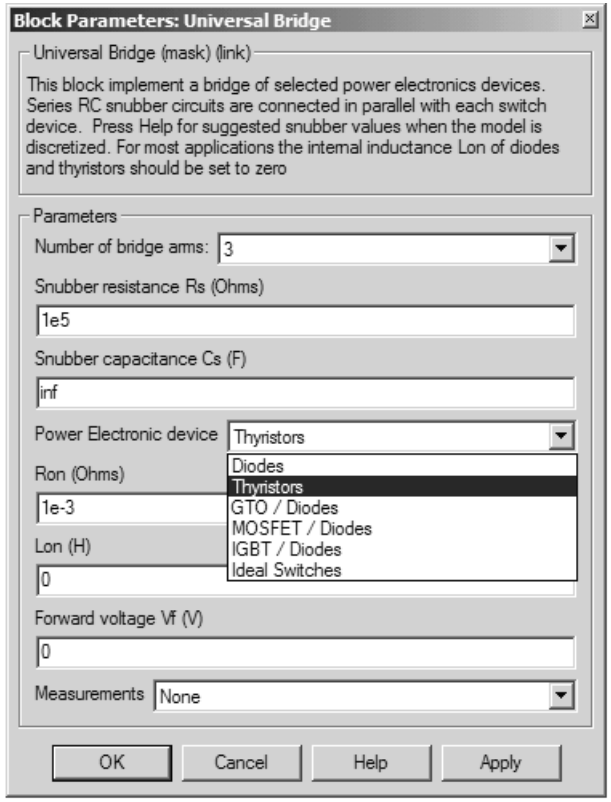


Рис. 11.51. Окно установки параметров универсального мостового модуля

На рис. 11.52 дан пример моделирования преобразователя переменного напряжения в постоянное, которое затем вновь преобразуется в переменное напряжение. В этом преобразователе использованы универсальные полумостовые модули.

Интересная деталь этого примера – возможность анализа спектра выходного сигнала преобразователя. Окно анализатора спектра (рис. 11.53) вызывается нажатием пиктограммы, расположенной слева под блоком осциллографов Scope2.

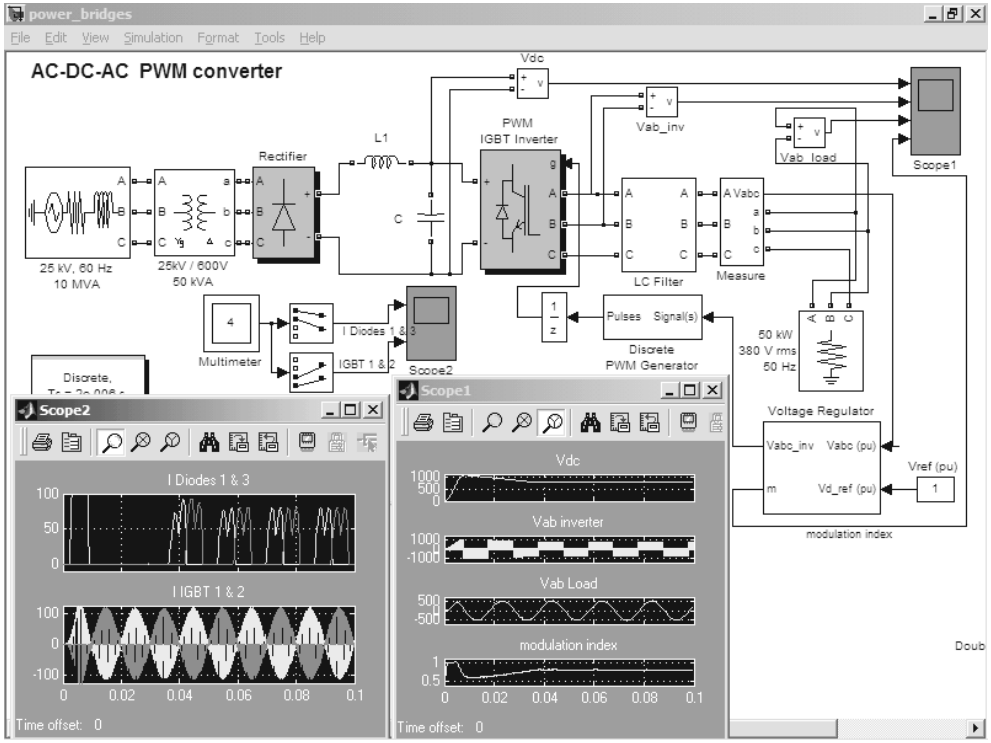


*Рис. 11.52. Пример преобразователя на основе блока **Universal Bridge***

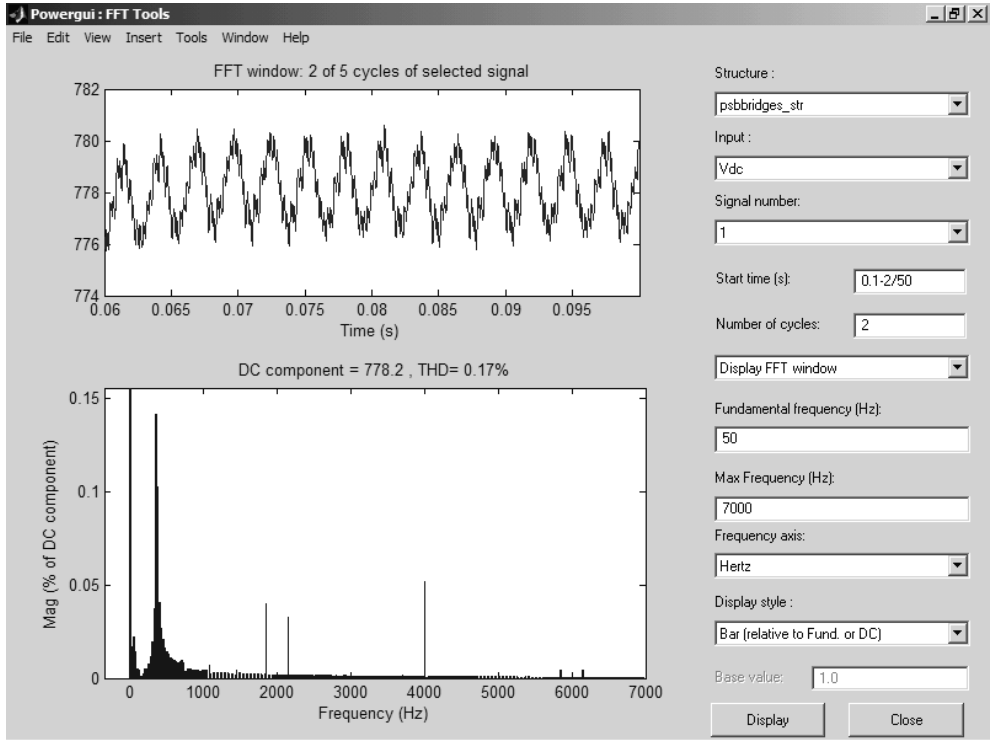


Рис. 11.53. Окно анализатора спектра

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.

11.5. Моделирование приводов электрических машин

11.5.1. Библиотека блоков электрических машин

Библиотека Mashines пакета SimPowerSystems Blockset (рис. 11.54) содержит модели ряда электрических машин постоянного тока, а также синхронных и асинхронных машин переменного тока. Имеются также блоки турбин, блоки стабилизации и блок измерения. Блоки позволяют создавать практически любые системы электропривода. Состав подобных блоков заметно расширен в новейшей реализации 4.* пакета, и новые блоки включены в отдельную библиотеку Application Library, которая описана в конце этого урока.

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 пару практических примеров моделирования, вошедших в библиотеку справочных материалов по пакету SimPowerSystems Blockset.

11.5.2. Пример моделирования привода двигателя постоянного тока

Рисунок 11.55 показывает простую систему управления двигателем постоянного тока – блок DC_Mashine с внешним возбуждением. Двигатель форсированно запускается на стадии его разгона с помощью импульсного регулятора постоянного

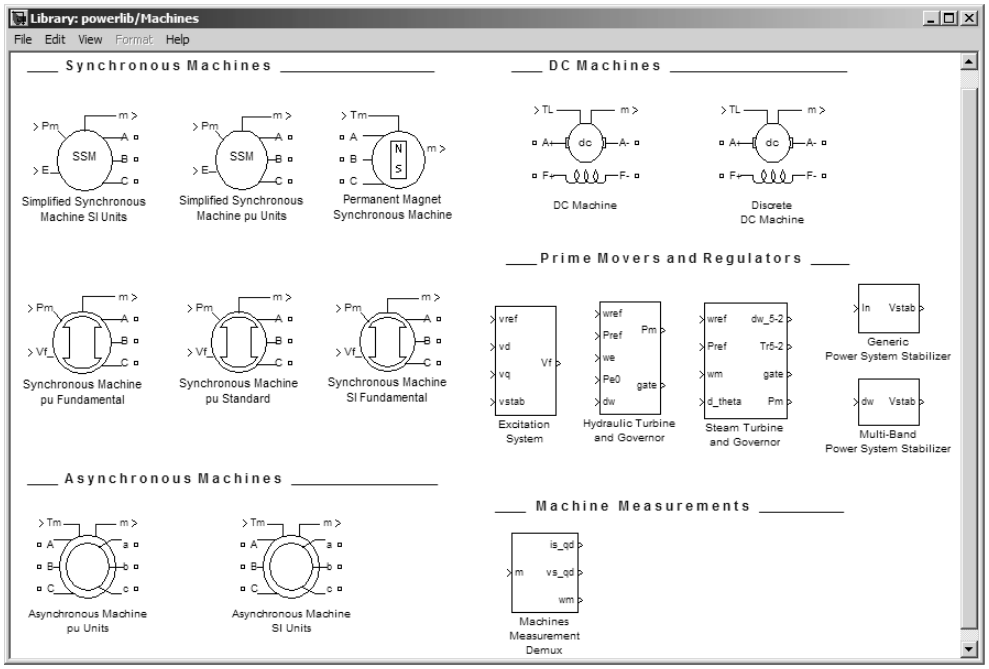


Рис. 11.54. Окно библиотеки Mashines

тока. Число оборотов двигателя контролируется выходом m. Можно заметить, что число оборотов растет во времени линейно и после небольшого перерегулирования устанавливается неизменным.

Окно установки параметров электродвигателя постоянного тока показано на рис. 11.56. Для управления двигателем применяется преобразователь на коммутирующем приборе Gto. С помощью переключателей можно выбрать различные режимы работы двигателя и провести моделирование для них.

11.5.3. Пример моделирования мощной синхронной машины

На рис. 11.57 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).

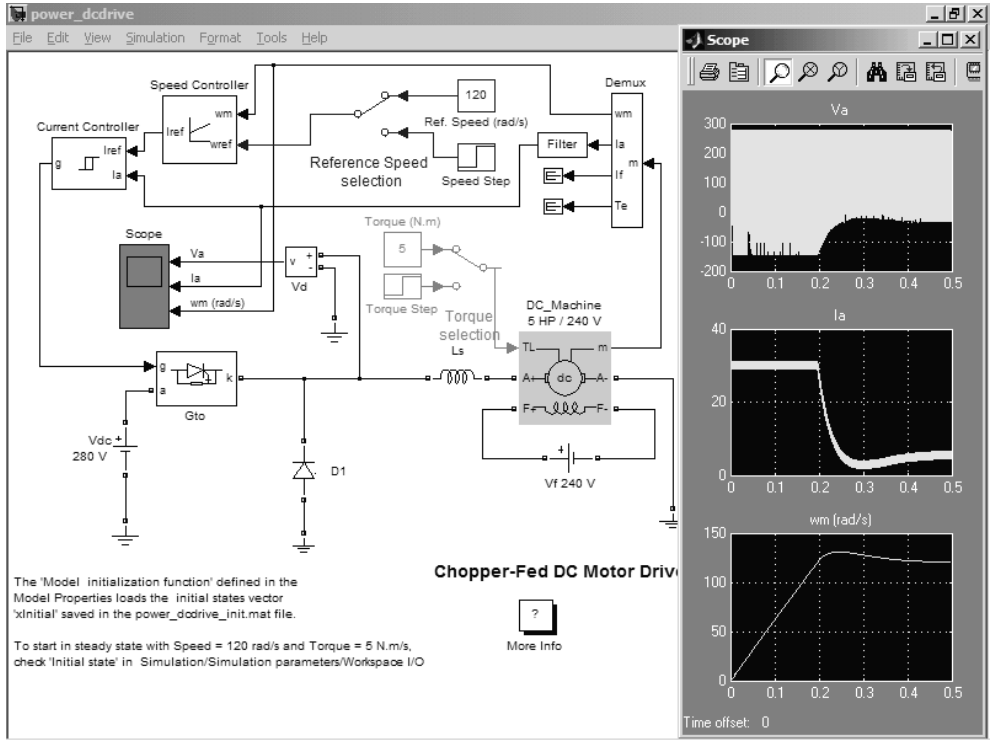


Рис. 11.55. Модель системы управления двигателем постоянного тока

С помощью трансформатора напряжение сети повышается до 230 кВ. С подсистемами этой модели заинтересованный читатель может ознакомиться самостоятельно.

11.5.4. Пример моделирования привода асинхронного двигателя

Пример моделирования асинхронного двигателя, питаемого от трехфазной сети, созданной импульсным преобразователем релейного типа, показан на рис. 11.58. Осциллограммы характеризуют работу системы на этапе разгона двигателя.

Следует отметить, что асинхронные машины в пакете SimPowerSystem Blockset представлены двумя моделями: **pu Units 1** (с трехфазным входом) и **pu Units** (со входом, имеющим две фазы с нейтралью).

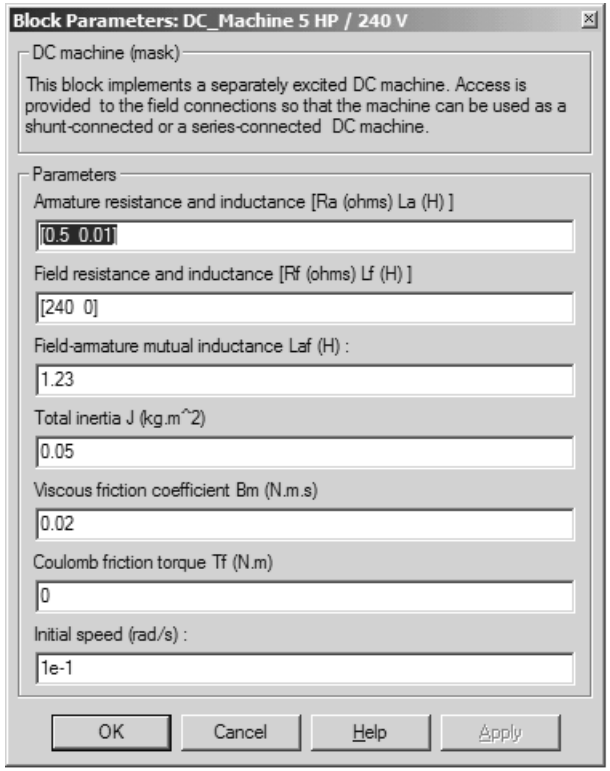


Рис. 11.56. Окно установки параметров блока ***DC_Mashine***

11.6. Моделирование электрических преобразователей электроэнергии

11.6.1. Моделирование импульсного преобразователя с ключом на полевом транзисторе

Преобразователи с импульсным регулированием обычно используют для управления силовым ключом широтно-импульсный модулятор. Рисунок 11.59 показывает одну из реализаций ШИМ, а также управляющий треугольный сигнал на его входе и выходные импульсы модулятора.

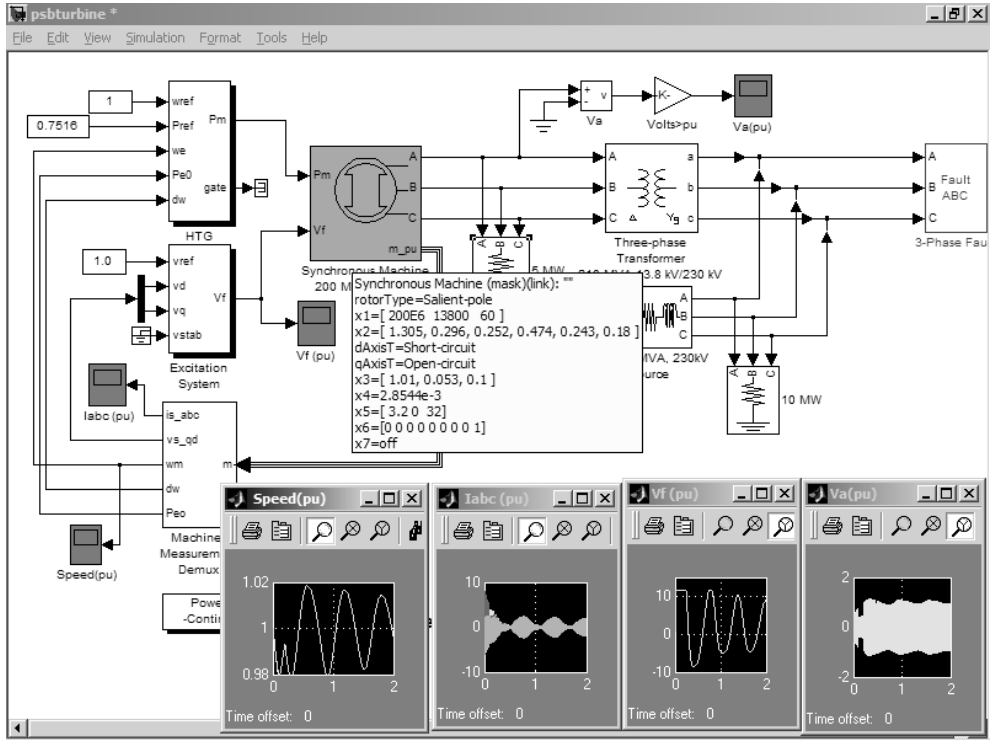


Рис. 11.57. Пример моделирования мощной синхронной машины

Обратите внимание на то, что все пояснения на этой модели даны на русском языке, поскольку модель не является встроенной в библиотеку примеров системы MATLAB.

На рис. 11.60 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. 11.60, 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 ди-

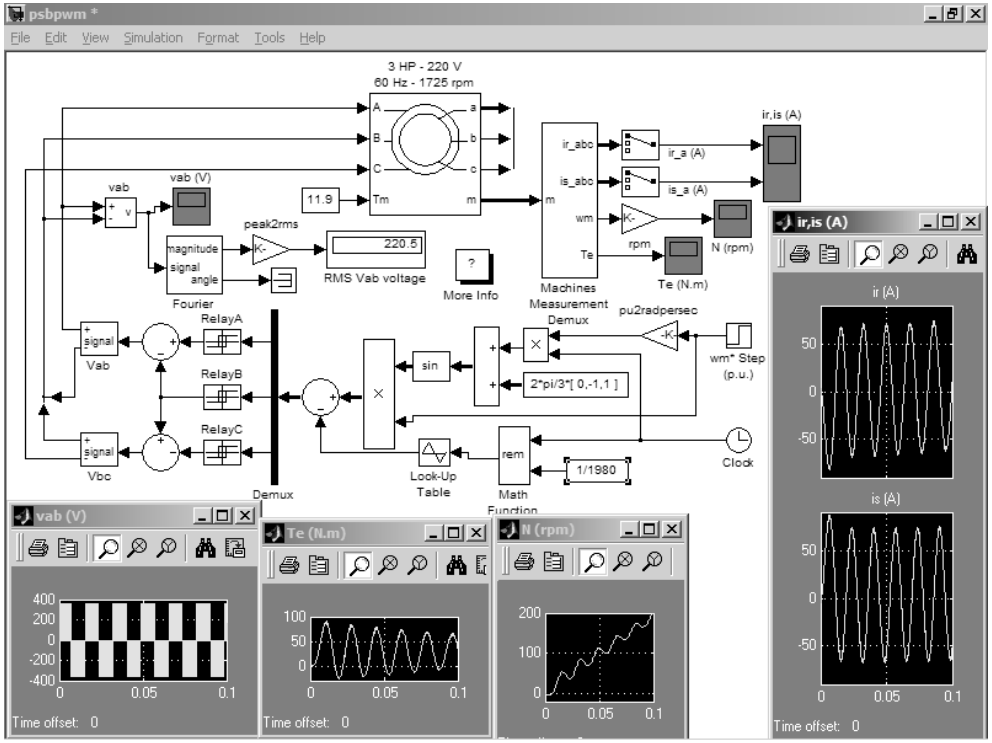


Рис. 11.58. Моделирование привода асинхронного двигателя, питаемого от трехфазного импульсного источника

намических 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.

11.6.2. Моделирование неуправляемых однофазных выпрямителей

Пакет расширения SimPowerSystem открывает обширные возможности в изучении путем моделирования различных преобразователей электрической энергии. Начнем с простого преобразователя энергии переменного тока в постоянный – мостового неуправляемого однофазного выпрямителя. Две равноценные схемы такого выпрямителя, широко применяемые в бытовой технике, представлены на рис. 11.61.

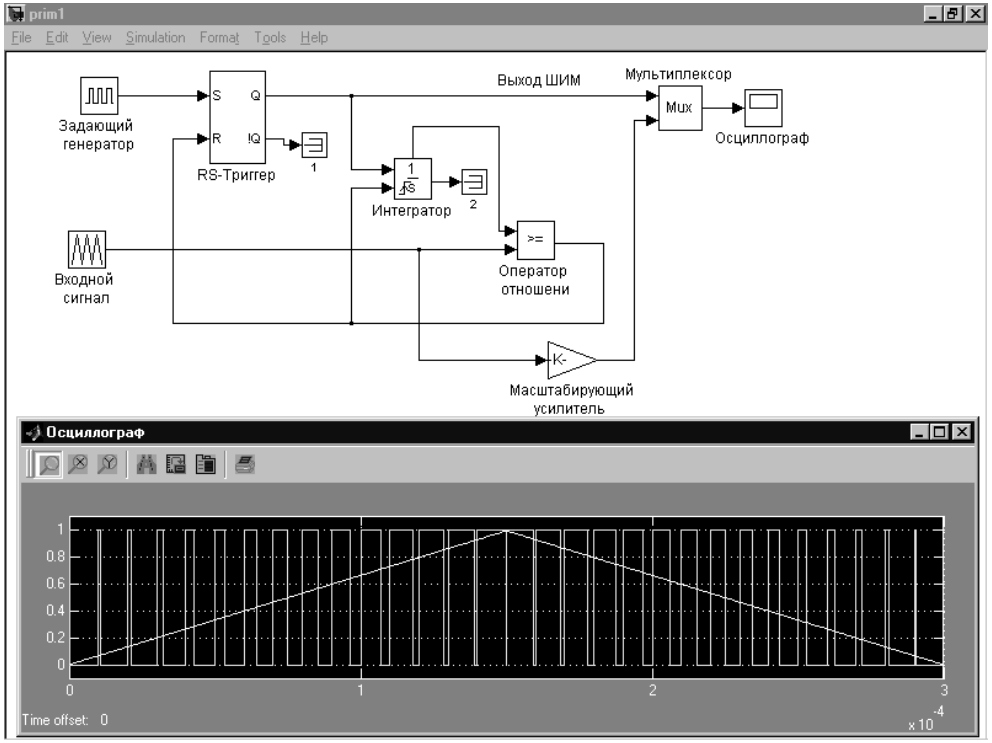


Рис. 11.59. Подсистема ШИМ

Оба выпрямителя преобразуют напряжение сети переменного тока 120 В 60 Гц в постоянный ток в нагрузке 5 Ом. Для сглаживания пульсаций используется LC-фильтр. Обе схемы на рис. 11.61 идентичны и отличаются реализацией модели выпрямителя: в первой схеме используется мост из 4 диодов, в другой – готовая модель такого моста на основе универсального блока **Universal Bridge**. Осциллограммы работы устройств подтверждают их идентичность (на верхней осциллограмме две кривые сливаются).

11.6.3. Моделирование трехфазных выпрямителей

Более эффективны широко применяемые в промышленности трехфазные выпрямители. Пример моделирования такого неуправляемого диодного выпрямителя представлен на рис. 11.62. Здесь в качестве фильтра используется катушка индуктивности, включенная последовательно с нагрузкой. Емкостный фильтр отсутствует, но, несмотря на это, пульсации тока в нагрузке достаточно малы.

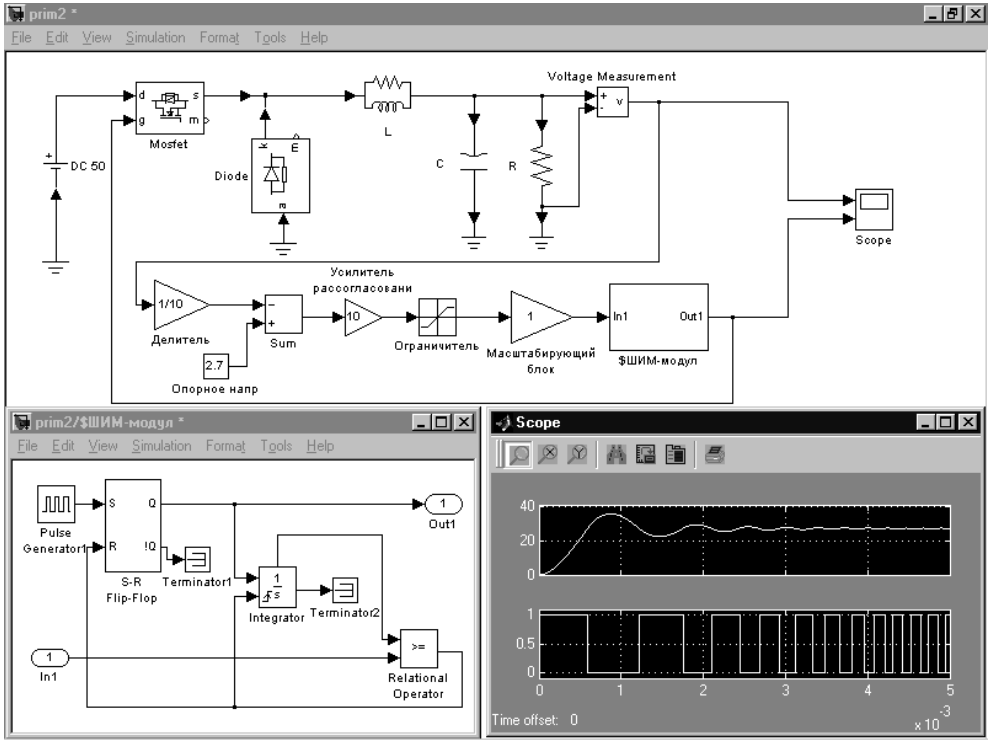


Рис. 11.60. Моделирование преобразователя с идеальным ключом, управляемым ШИМ

11.6.4. Моделирование однофазных инверторов

Как известно, наибольший коэффициент полезного действия обеспечивают импульсные преобразователи напряжения, именуемые также инверторами. Ныне такие преобразователи применяются повсеместно в блоках питания бытовой и сложной измерительной и вычислительной аппаратуры.

Принцип построения трех простейших таких устройств поясняет рис. 11.63. В левой части модели представлены идеализированные расчетные выражения, позволяющие сравнивать эти устройства. В подробном описании данные устройства не нуждаются – те, кому это описание все же необходимо, могут найти его в учебной литературе по преобразовательной технике. Обратите, однако, внимание на то, что в первом преобразователе постоянного напряжения в постоянный ток нагрузка имеет источник э.д.с., то есть преобразователь моделирует работу зарядного устройства. Два других преобразователя преобразуют постоянное напряжение в переменное.

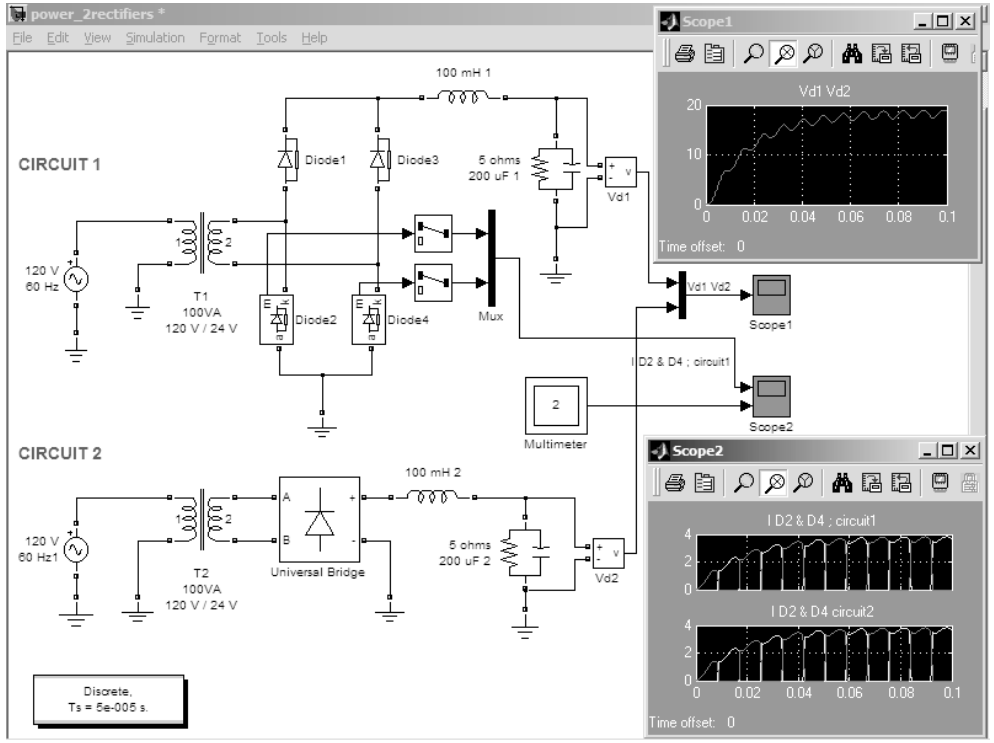


Рис. 11.61. Моделирование однофазных мостовых выпрямителей

Осциллограммы напряжений этих преобразователей (слева направо для схем рис. 11.63 сверху вниз) представлены на рис. 11.64. Нетрудно заметить, что характер преобразований определяется характером изменения ширины импульсов, управляющих ключами.

11.6.5. Моделирование трехфазных инверторов

Пример моделирования трехфазного трехуровневого управляемого выпрямителя (преобразователя) показан на рис. 11.65. Модель демонстрирует переходные процессы после включения преобразователя и затем после подключения к нагрузке дополнительной нагрузки. В этом устройстве используется управляемый трехуровневый мост.

В настоящее время широко используются различные устройства, например электромобили, которые имеют источник питания постоянного тока, но исполнительные элементы на основе асинхронных или синхронных двигателей. Они нуждаются в преобразователях энергии постоянного тока в трехфазный переменный

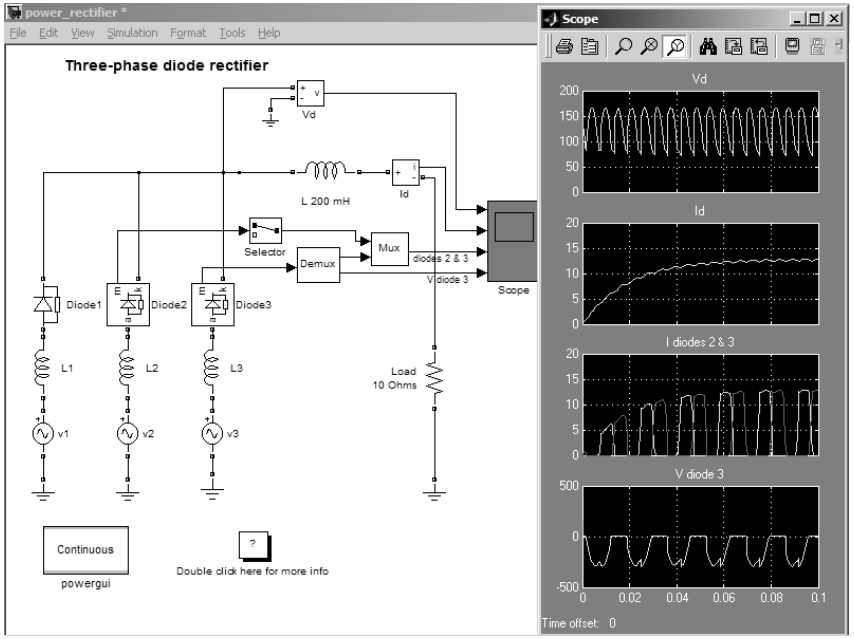


Рис. 11.62. Моделирование однофазных мостовых выпрямителей

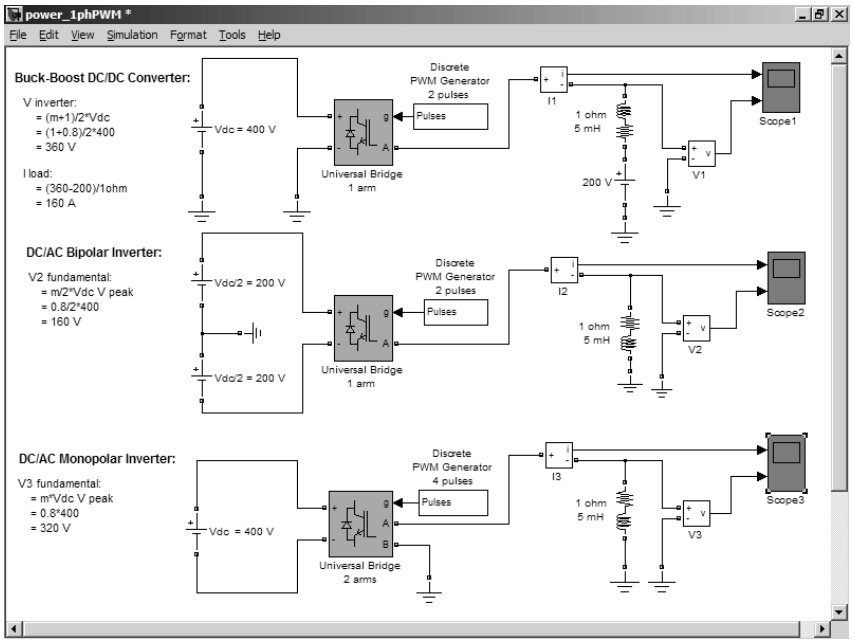


Рис. 11.63. Моделирование трех простейших импульсных преобразователей

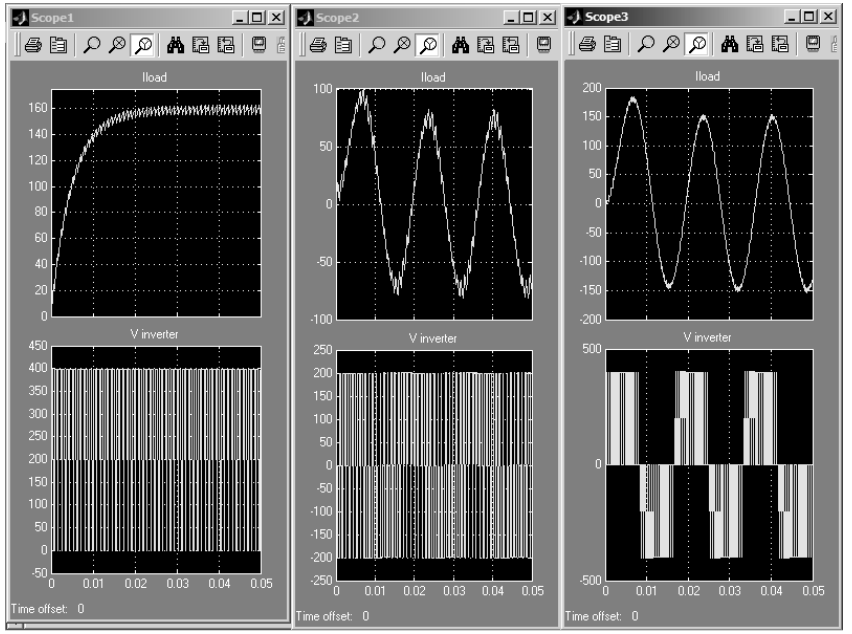


Рис. 11.64. Осциллограммы напряжения и тока инвертора для трех схем рис. 11.90

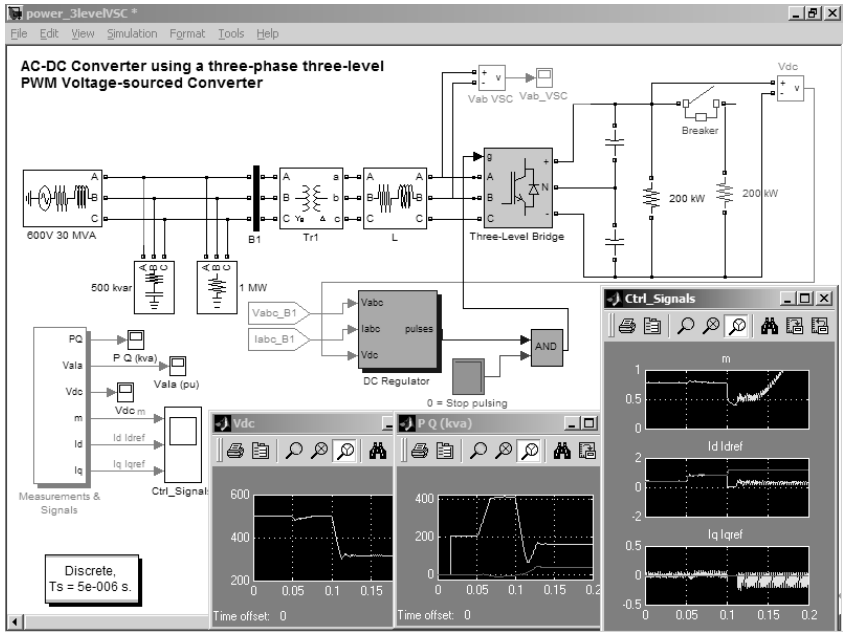


Рис. 11.65. Модель мощного трехфазного управляемого выпрямителя (преобразователя)

ток. Рисунок 11.66 представляет модели двух таких устройств. Обе модели построены на основе универсального моста с двухуровневой импульсной коммутацией. Осциллограммы поясняют работу этих устройств.

Еще один пример такого устройства на основе универсального моста с трехуровневой коммутацией представлен на рис. 11.67. Повышение числа уровней коммутации способствует получению более чистого трехфазного синусоидального напряжения.

11.6.6. Пример моделирования многоимпульсного GTO-преобразователя

Еще лучшие результаты могут быть получены при использовании многоимпульсных GTO-преобразователей. На рис. 11.68 показан преобразователь с 48-импульсным GTO-инвертором. Осциллограммы иллюстрируют раскачку инвертора и переходный процесс после включения мощной нагрузки. Нетрудно заметить, что установление стационарного режима происходит за доли одного периода трехфазного напряжения.

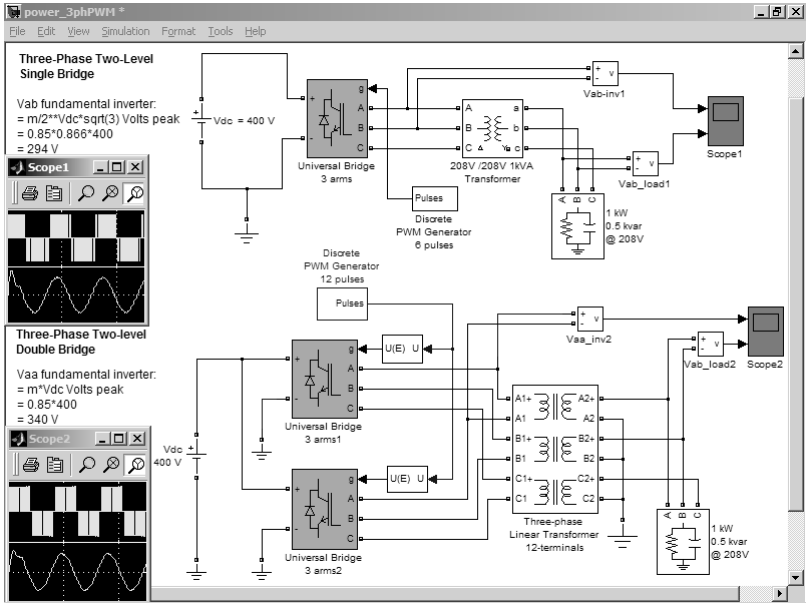


Рис. 11.66. Две модели преобразователей постоянного напряжения в трехфазное переменное напряжение

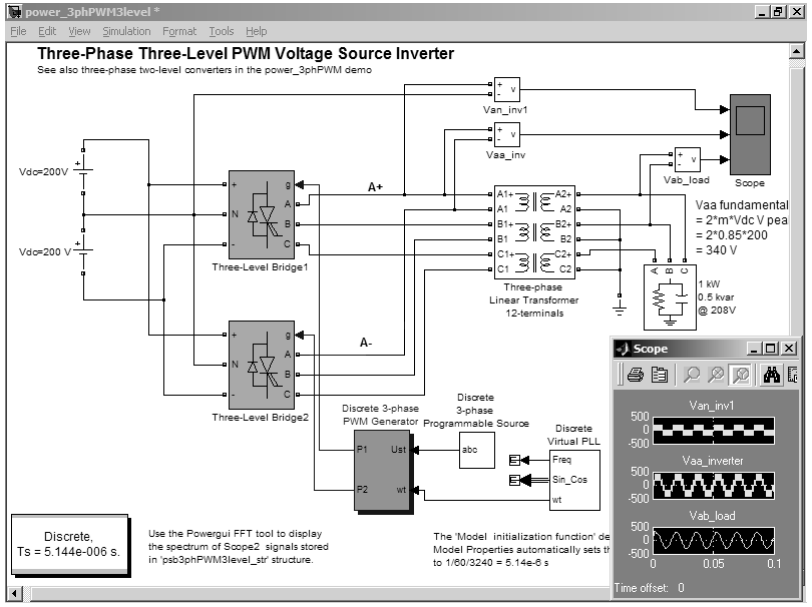


Рис. 11.67. Модель преобразователя постоянного напряжения в трехфазное переменное напряжение с трехуровневой коммутацией

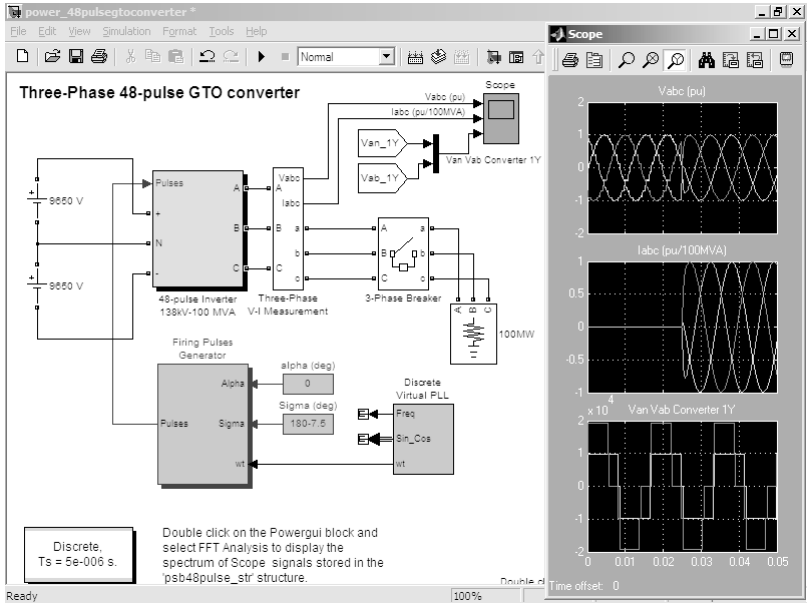


Рис. 11.68. Модель преобразователя постоянного напряжения в трехфазное переменное напряжение с 48-импульсным GTO-инвертором

11.6.7. Моделирование трехфазного инвертора для асинхронных двигателей

Мы уже отмечали возможность применения трехфазных инверторов для асинхронных двигателей – рис. 11.58. Рисунок 11.69 показывает еще один пример такого рода. Здесь для питания асинхронного электродвигателя используется PWM-инвертор.

11.6.8. Моделирование динамической нагрузки и управляемого трехфазного источника

В состав блоков пакета SimPowerSystems входят управляемый трехфазный источник и динамическая нагрузка, позволяющая проверять его динамические свойства. Рисунок 11.70 демонстрирует простую электротехническую систему, в которой объединены эти два блока и обеспечен контроль над их работой.

В модель входят два виртуальных осциллографа. Осциллограммы одного из них представлены на рис. 11.70 под диаграммой модели. Осциллограммы другого приведены на рис. 11.71. Любопытно проявление на них узоров (темных ром-

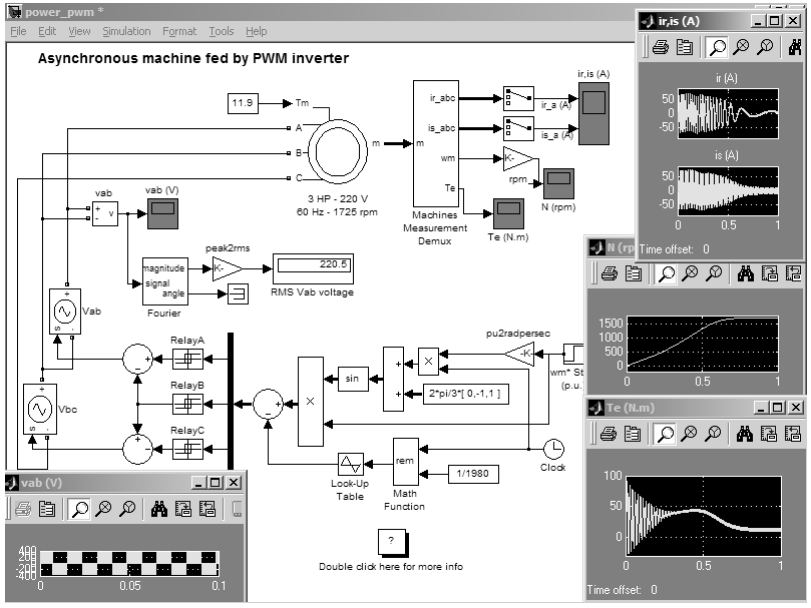


Рис. 11.69. Модель питания асинхронного двигателя от преобразователя постоянного напряжения в трехфазное переменное напряжение с трехуровневой коммутацией

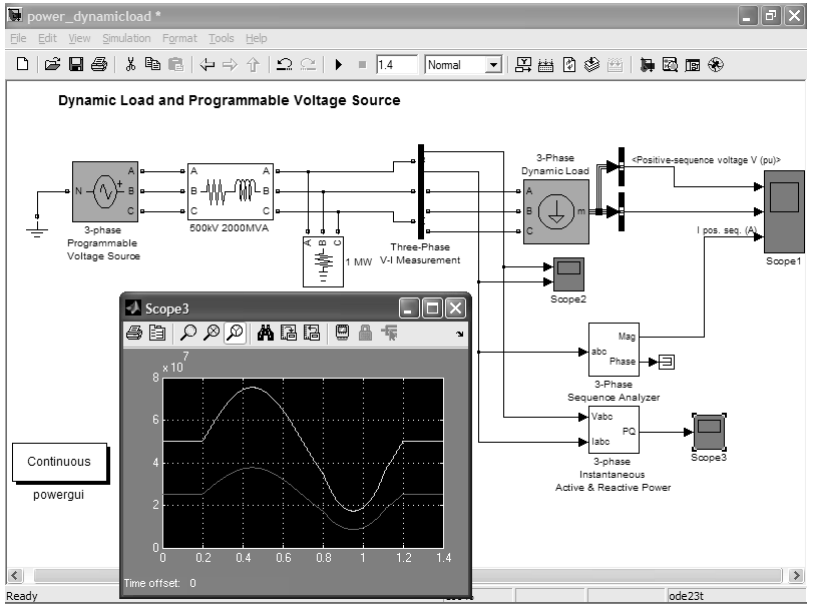


Рис. 11.70. Модель системы с управляемым трехфазным источником и динамической нагрузкой

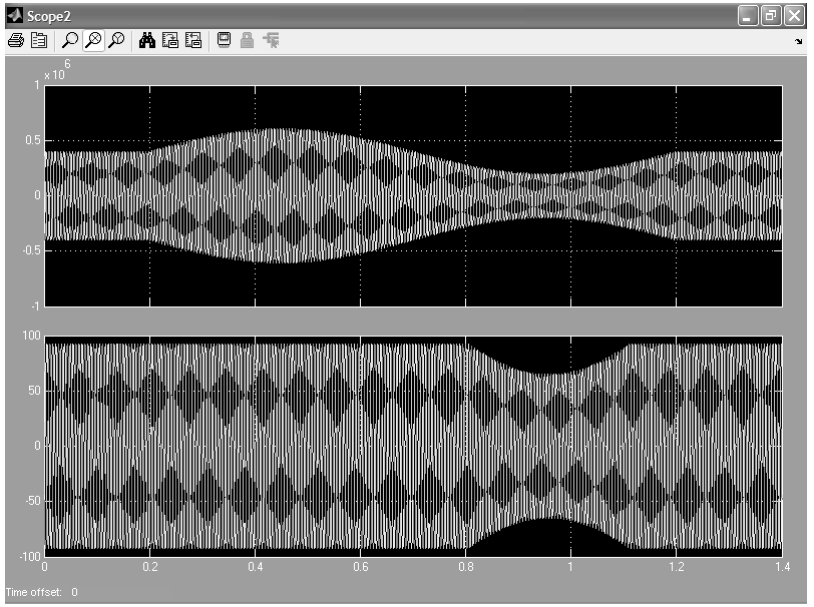


Рис. 11.71. Осциллограммы виртуального осциллографа Scope 1 модели рис. 11.70

биков), вызванное стробоскопическим эффектом (возможно, и проявлением муара).

На примере модели рис. 11.70 еще раз отметим особенности блока powergui. Он представлен на рис. 1.70 в левом нижнем углу. Активизация пиктограммы этого блока вызывает появление окна блока, частично показанного на рис. 1.71 слева (мы не раз приводили ранее вид окна этого блока в полном виде). Основное место на рис. 11.72 занимает большое окно параметров линии передачи. Из него видно, что используется 734-киловольтная трехфазная линия передачи. Мы иллюстрируем лишь доступ к одному из ряда окон блока powergui. Установка и наблюдение параметров модели с помощью этого блока представляют интерес для специалистов в области передачи электроэнергии на большие расстояния.

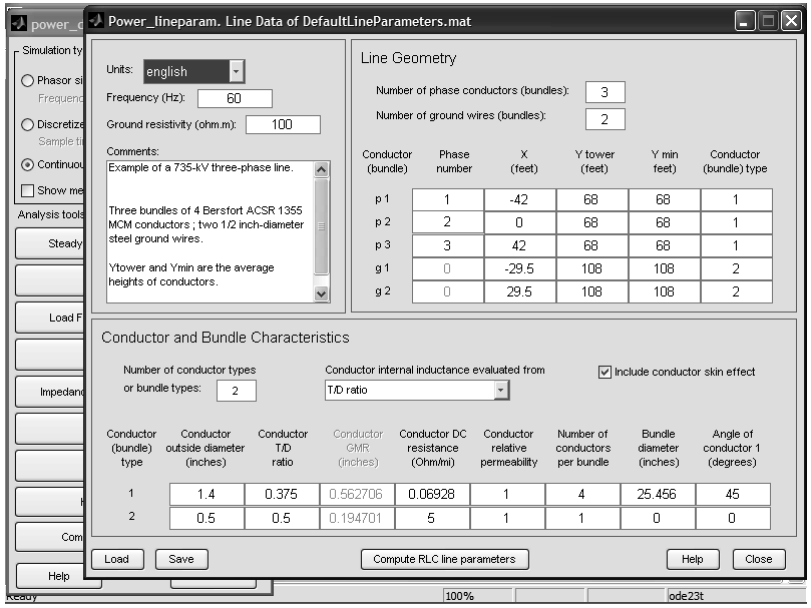


Рис. 11.72. Пример работы с GUI-моделью рис. 11.70

11.7. Новая библиотека Application Library в SimPowerSystems 4.*

11.7.1. Состав библиотеки Application Library

Новая библиотека Application Library появилась в версиях SimPowerSystems 4.*. Она значительно расширяет набор средств моделирования сложных энергетических и электромеханических систем и устройств. Библиотека состоит из трех разделов:

- **Distributed Resource Library** – библиотека средств распределения энергии (турбины ветровых электростанций);
- **Electric Drive Library** – библиотека электрических устройств (машины постоянного и переменного тока, валы и редукторы);
- **Flexible AC Transmission Systems (FACTS) Library** – гибкие системы трансмиссии переменного тока.

Для ознакомления с возможностями новой библиотеки в разделе **Demos** справки MATLAB есть ряд примеров.

11.7.2. Блоки турбин ветровых электростанций

Раздел описываемой библиотеки Distributed Resource Library содержит всего три блока. Они представлены на рис. 11.73.

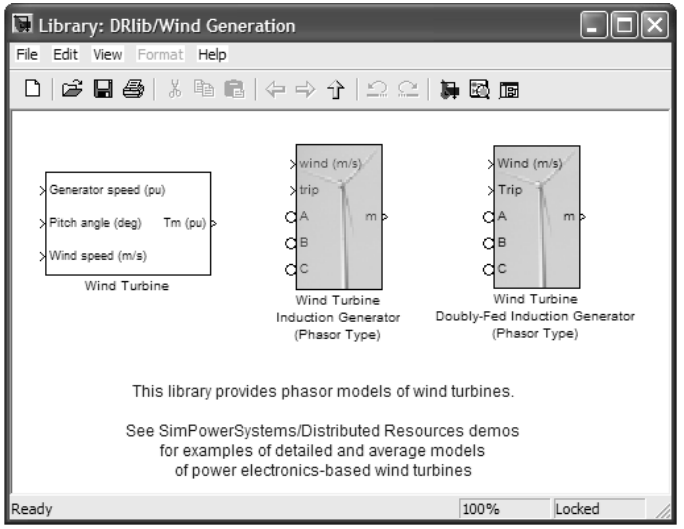


Рис. 11.73. Состав блоков библиотеки турбин ветровых электростанций

Поскольку погода обычно неустойчива, то наличие или отсутствие ветра – дело случая. Поэтому ветровые электростанции сами по себе не способны обеспечить бесперебойное снабжение электроэнергией потребителей. Они работают, как правило, в составе распределенных систем электроснабжения. Достаточно мощные ветровые электростанции могут повышать отдачу электроэнергии, будучи включенными в состав распределенной системы. Пример такого рода системы представлен на рис. 11.74. В системе использована модель 9-мегаватной ветровой электростанции (турбины).

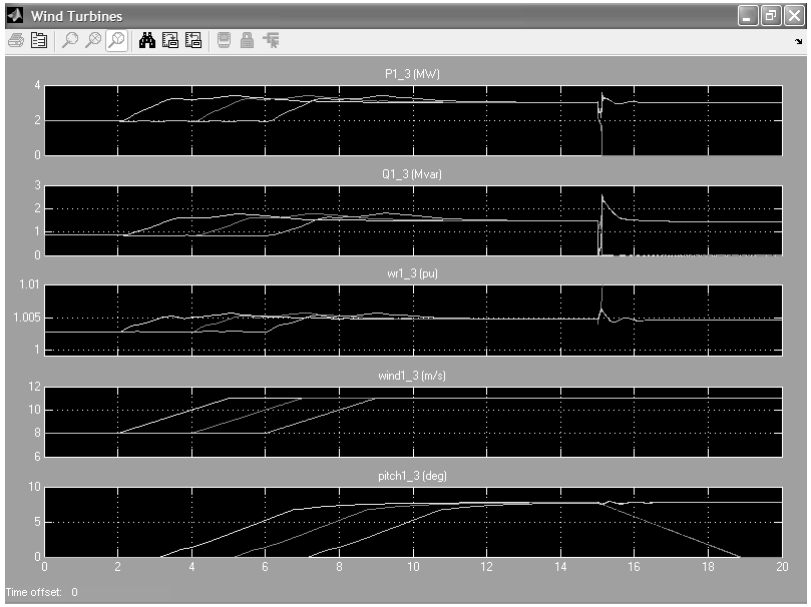


Рис. 11.74. Пример моделирования системы с 9-мегаватной ветровой электростанцией

О сложности электромеханических процессов, происходящих при работе этой системы, наглядно свидетельствуют осциллограммы, представленные на рис. 11.75.

11.7.3. Блоки электрических устройств

Библиотека **Electric Drive Library** содержит обширный набор электрических устройств, разбитых на три группы (рис. 11.76):

- **DC Drive** – машины постоянного тока;
- **AC Drive** – машины переменного тока;
- **Shafts and speed reducers** – библиотека блока валов и редукторов.

11.7.4. Блоки машин постоянного тока

Блоки машин постоянного тока представлены в окне, показанном на рис. 11.77. Назначение блоков вполне очевидно из их названий. Всего предложено 7 блоков, дающих достаточно представительный набор основных классов таких машин. Подробное описание математических и физических моделей этих блоков в рамках объема данного урока не представляется возможным. Заинтересованный пользователь может познакомиться с ним по справке или по документации на описываемый пакет расширения.

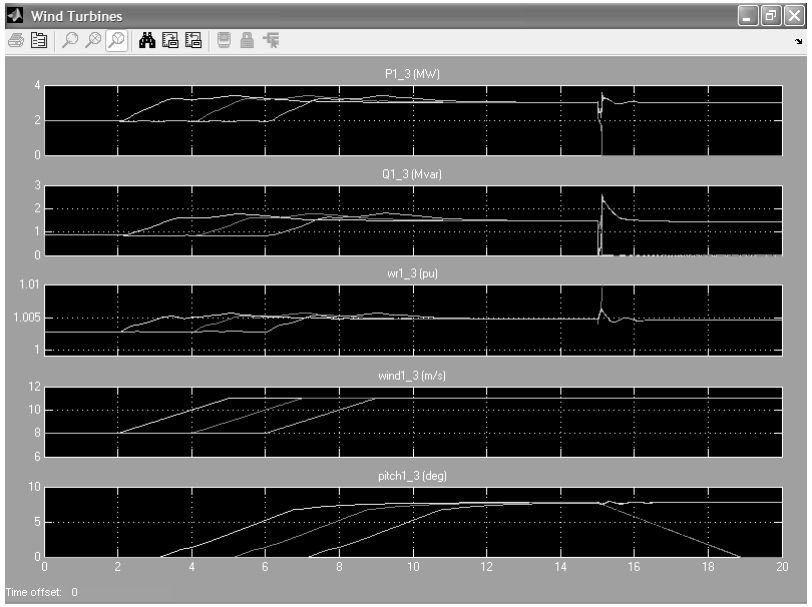


Рис. 11.75. Осциллограммы процессов, происходящих при моделировании системы рис. 1.74

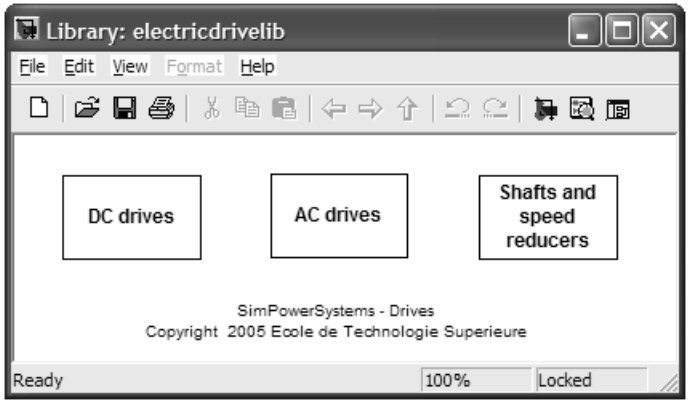


*Рис. 11.76. Окно с разделами библиотеки **Electric Drive Library***

В разделе справки **Demos** можно найти множество примеров на применение этих блоков, хорошо раскрывающих сущность моделирования систем с машинами постоянного тока. Ограничимся двумя из этих примеров. Блок-диаграмма одного из них показана на рис. 11.78. Эта блок-диаграмма моделирует запуск выпрямительного однофазного двигателя постоянного тока.

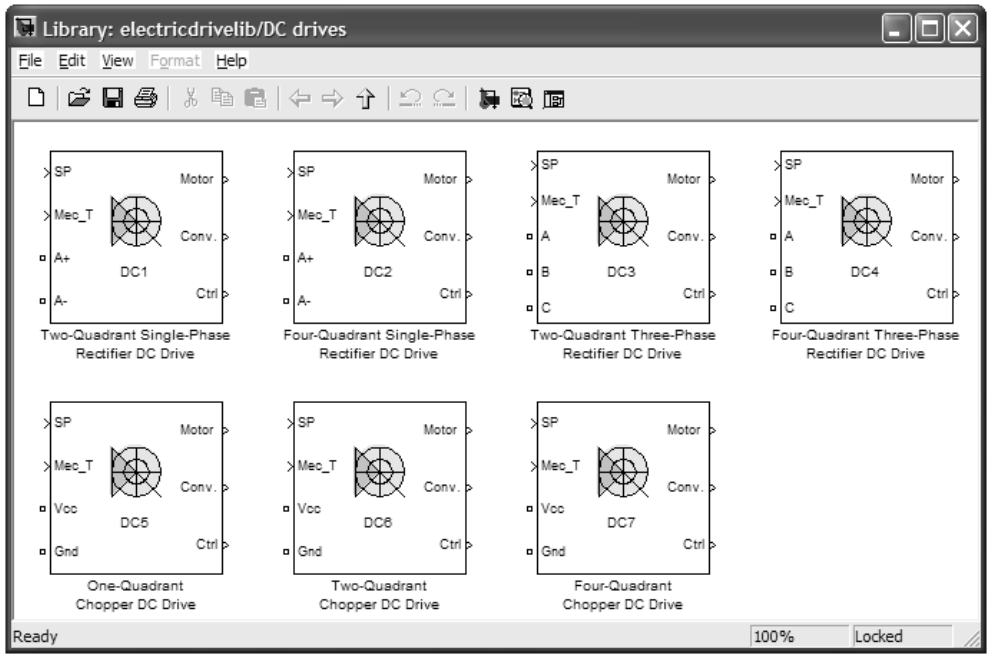


Рис. 11.77. Окно с блоками машин постоянного тока

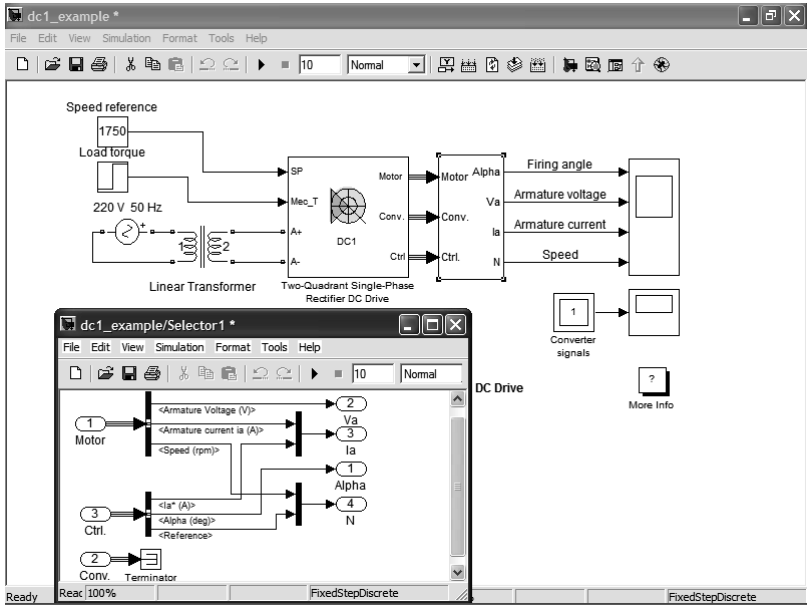


Рис. 11.78. Пример моделирования системы с двигателем постоянного тока DC1

На рис. 11.79 представлено окно задания параметров выпрямительного однофазного двигателя постоянного тока. Окно содержит три вкладки для установки параметров двигателя постоянного тока, конвертора и контроллера. Все эти устройства интегрированы в описанную модель машины. Осциллограммы работы данной модели представлены на рис. 11.80.

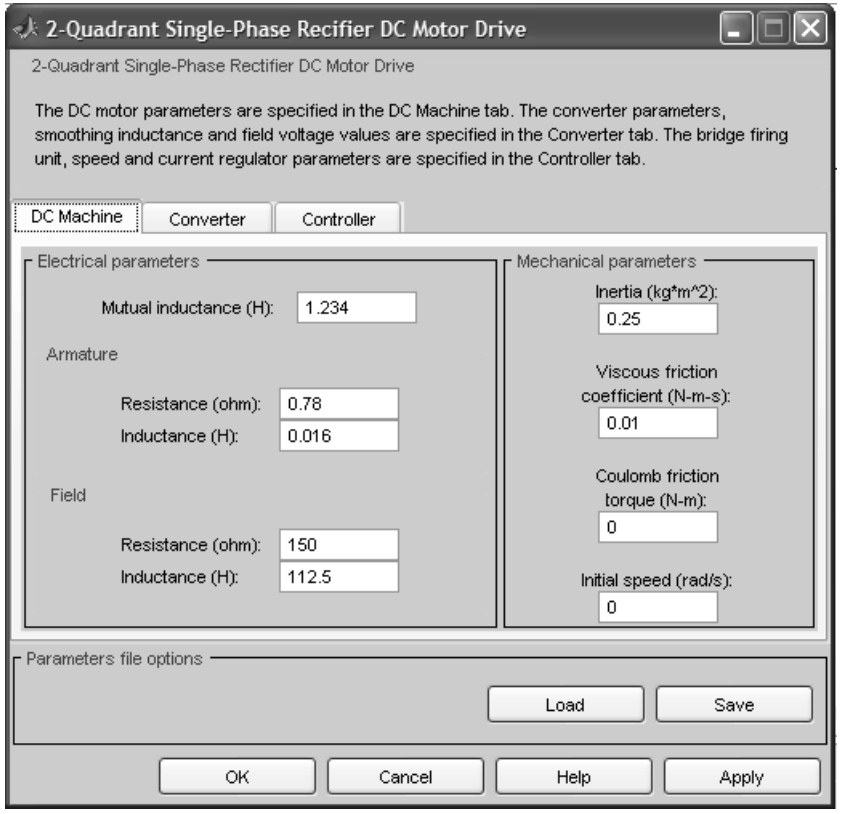


Рис. 11.79. Окно установки параметров выпрямительного однофазного двигателя постоянного тока

Блок-диаграмма другого примера, на моделирование системы с двигателем постоянного тока DCC3, представлена на рис. 11.81. На этот раз моделируется система с двухквадрантным двигателем с трехфазным выпрямителем. Осциллограммы работы данной модели представлены на рис. 11.82.

11.7.5. Блоки машин переменного тока

Блоки машин переменного тока представлены в окне, показанном на рис. 11.83. Здесь представлены 6 блоков, назначение которых вполне очевидно из их назва-

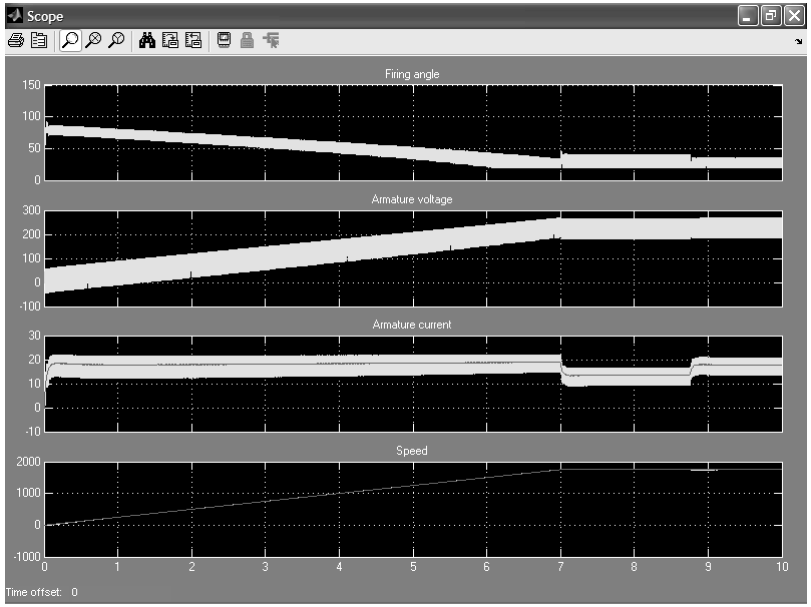


Рис. 11.80. Осциллограммы работы модели выпрямительного однофазного двигателя постоянного тока

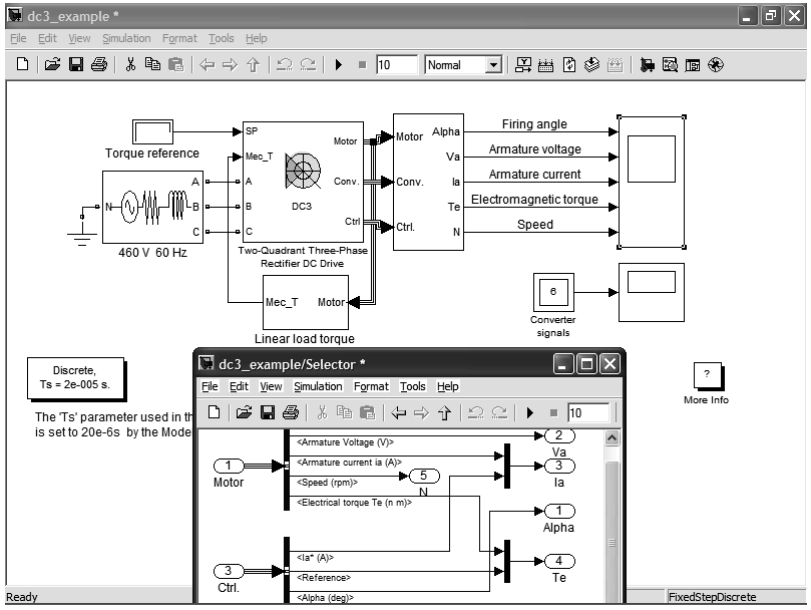


Рис. 11.81. Пример моделирования системы с двигателем постоянного тока DC3

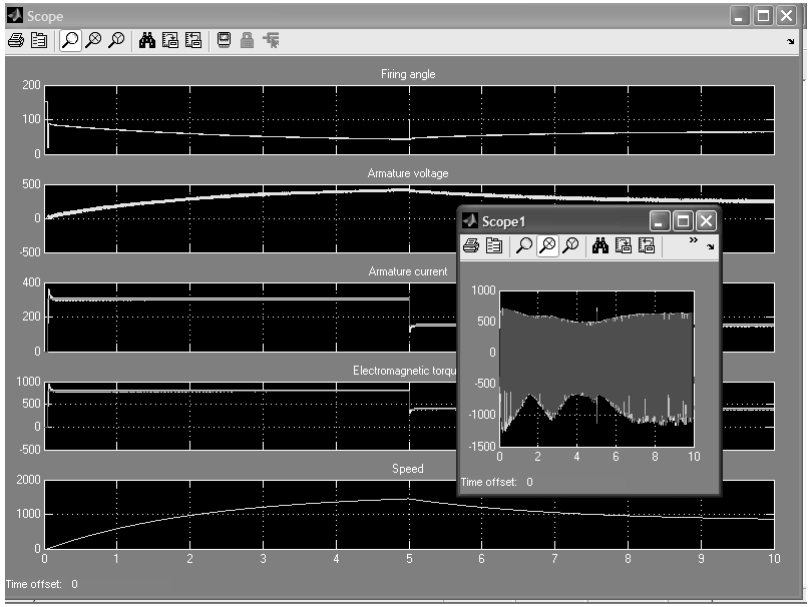


Рис. 11.82. Осциллограммы работы модели выпрямительного однофазного двигателя постоянного тока

ний (разумеется, для специалистов по проектированию систем электропитания и управления двигателями переменного тока).

На все блоки в разделе **Demos** справки есть интересные примеры применения, которые помогут заинтересованному читателю ознакомиться с техникой моделирования систем с машинами переменного тока. В связи с этим ограничимся примером, представленным на рис. 11.84. Здесь показана субмодель блока **demux**. Временные диаграммы работы модели представлены на рис. 11.85. Они характеризуют разгон двигателя после подачи на него напряжения питания.

11.7.6. Блоки валов и редукторов

Раздел библиотеки Shafts and speed reducers представлен всего двумя блоками – валов и редукторов. Они видны в окне этого раздела библиотеки – рис. 11.86.

Оба блока представлены в простых демонстрационных примерах. Один из них для блока **Shafts** (валы) представлен на рис. 11.87. Там же показаны окно установки параметров блока механического вала и осциллограммы работы представленной модели.

Другая модель, представленная на рис. 11.88, демонстрирует применение блока **Speed Reducer**. Тут также показаны окно установки параметров блока и осциллограммы работы модели.

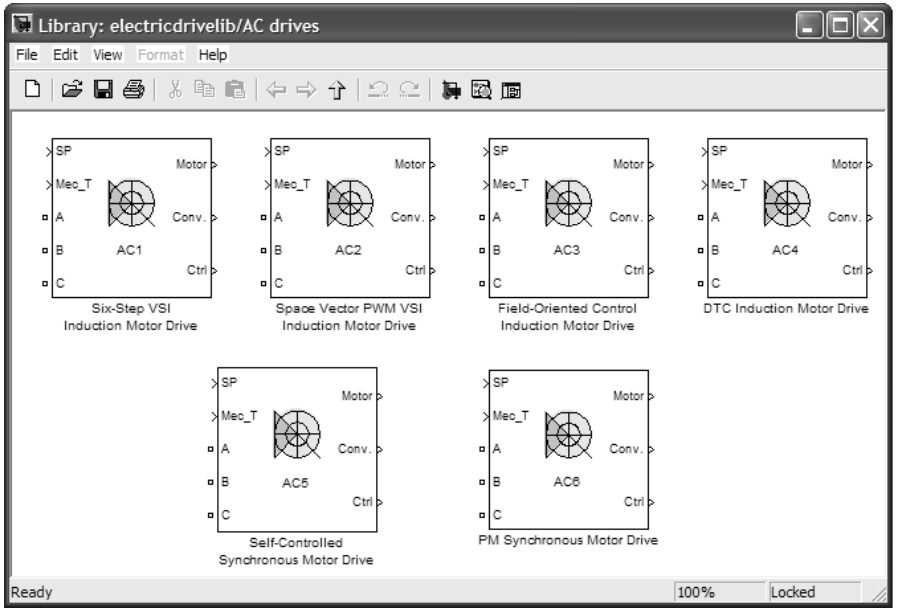


Рис. 11.83. Набор блоков машин переменного тока

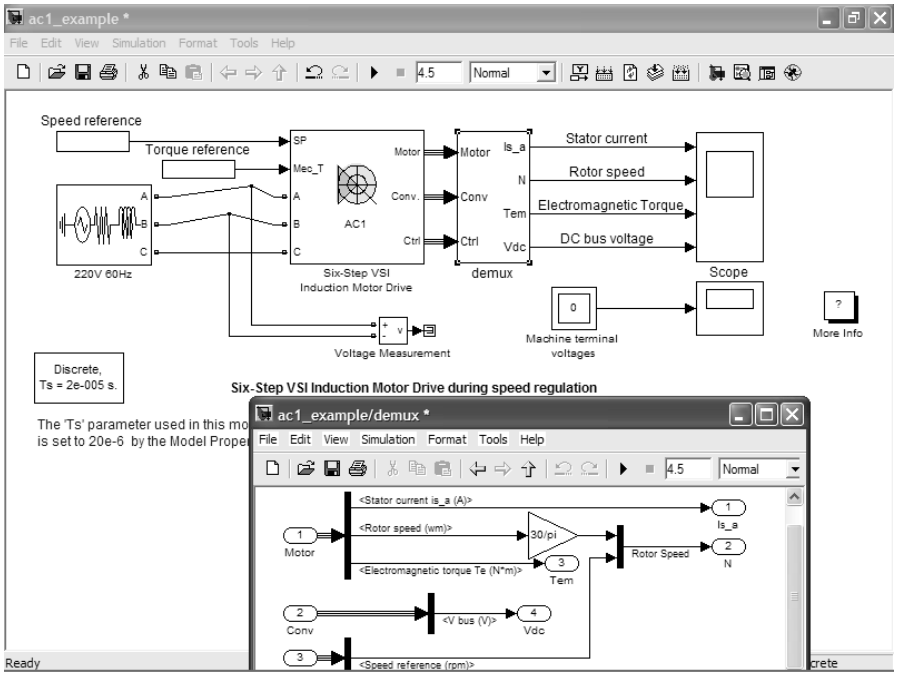


Рис. 11.84. Моделирование системы с машиной переменного тока AC1

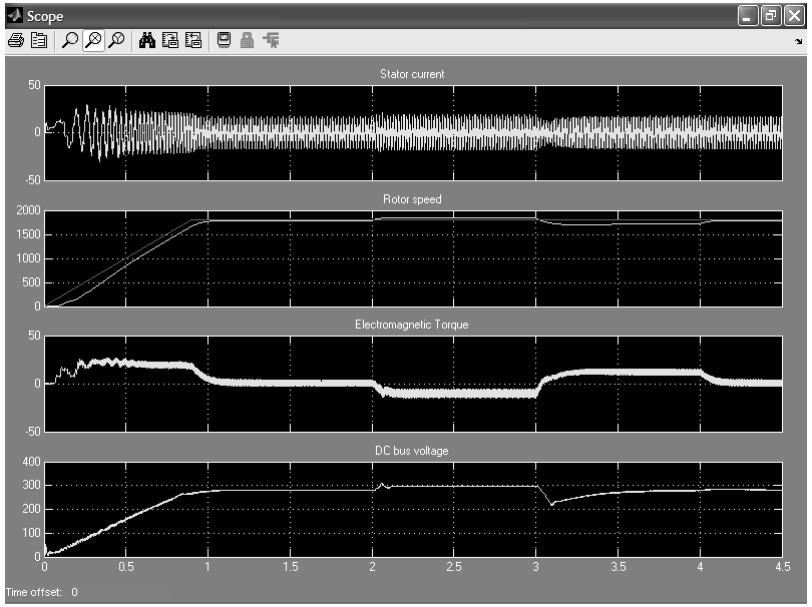


Рис. 11.85. Результаты моделирования системы с машиной переменного тока AC1

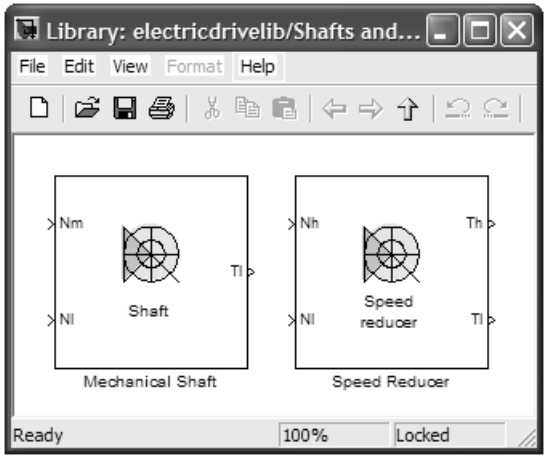


Рис. 11.86. Окно раздела библиотеки Shafts and speed reducers

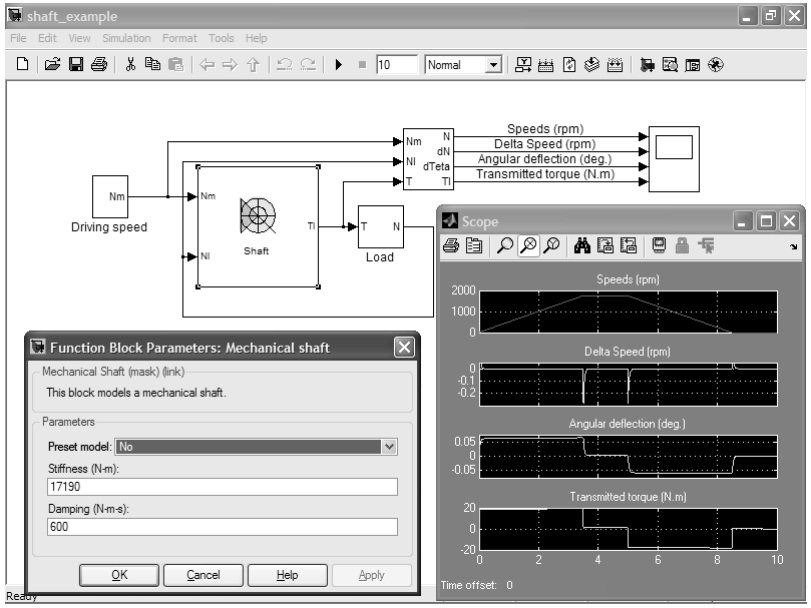


Рис. 11.87. Модель с блоком механического вала и результаты моделирования по ней

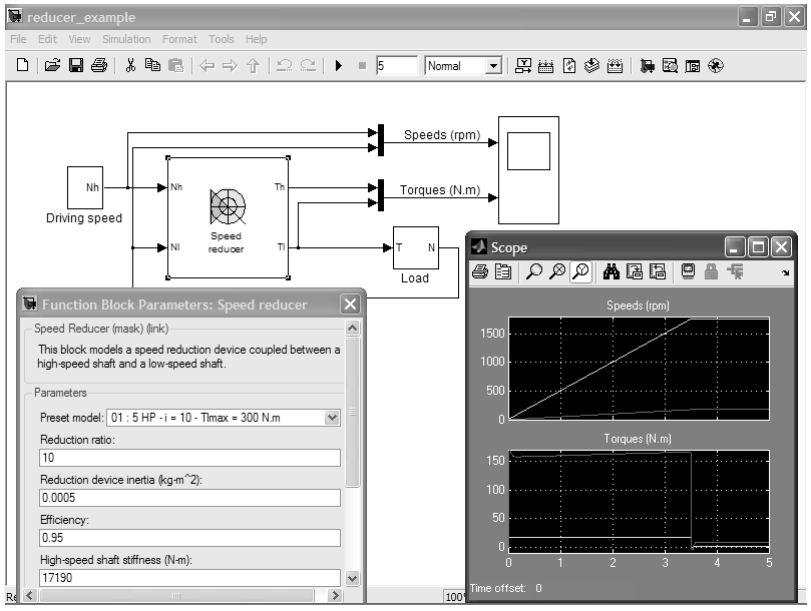


*Рис. 11.88. Модель с блоком **Speed Reducer** и результаты моделирования системы с ним*

11.7.7. Блоки библиотеки гибких систем передачи на переменном токе

Третий раздел библиотеки **Application Library** представлен тремя наборами блоков в окне, показанном на рис. 11.89:

- **Power Electronics Based FACTS** – блок просмотра демонстрационных примеров по моделированию основных гибких систем передачи на переменном токе FACTS (flexible AC transmission systems);
- **Transformers** – блоки преобразователей (два раздела по два блока в каждом);
- **HVDC Systems** – блок моделирования высоковольтных систем передачи постоянного тока HVDC (High-Voltage Direct Current). Всего четыре блока.

На применение блоков этой библиотеки в разделе справки **Demos** есть множество примеров. В большинстве своем это достаточно сложные примеры, понятные

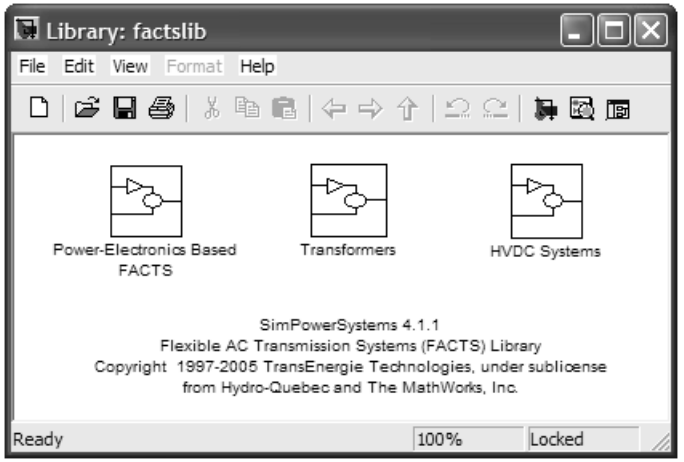


Рис. 11.89. Окно с блоками библиотеки гибких систем передачи на переменном токе

узким специалистам в области крупных энергетических систем. Надо полагать, что они в состоянии просмотреть эти примеры и сделать должные выводы об их полезности. В связи с этим рассмотрим один из достаточно простых примеров построения системы HVDC, модель которого представлена на рис. 11.90.

В этом примере источник переменного тока (500 кВ, 60 Гц, 5000 МВА) с помощью управляемого выпрямителя Rectifier преобразуется в постоянный ток, который передается по линии постоянного тока с длиной 300 км. С выхода линии сигнал подается на мощный импульсный инвертор, который превращает постоянное

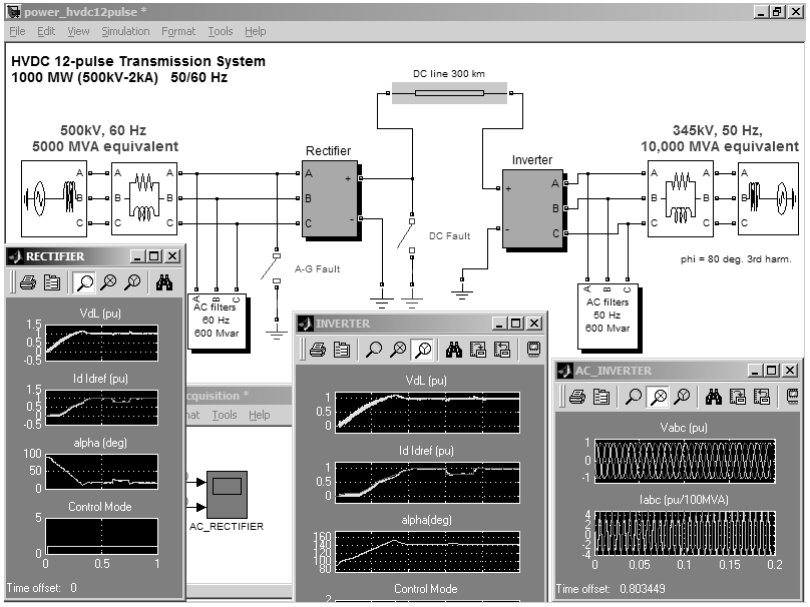


Рис. 11.90. Пример моделирования высоковольтной системы передачи электроэнергии на постоянном токе

напряжение в переменное (345 кВ, 50 Гц). Таким образом, данная энергосистема преобразует трехфазное напряжение с одной амплитудой и частотой в другое, также трехфазное напряжение с иными амплитудой и частотой. Передача энергии идет на постоянном токе, что заметно уменьшает ее потери. Модель позволяет исследовать сложные переходные процессы в этой системе, которые представлены осциллограммами виртуальных осциллографов, используемых для контроля переходных процессов в этой системе.

11.8. Другие библиотеки и примеры SimPowerSystems

11.8.1. Библиотека инструментов

Библиотека инструментов Measurements предназначена для предоставления самых разнообразных средств измерений. Ее окно показано на рис. 11.91 справа. В этом окне представлены пять измерительных блоков, назначение которых вполне очевидно. Это измерители тока, напряжения и импеданса (полного сопротивления), мультиметр и трехфазный измеритель напряжения и тока.

Внизу окна расположены три большие пиктограммы дополнительной Extra-библиотеки:

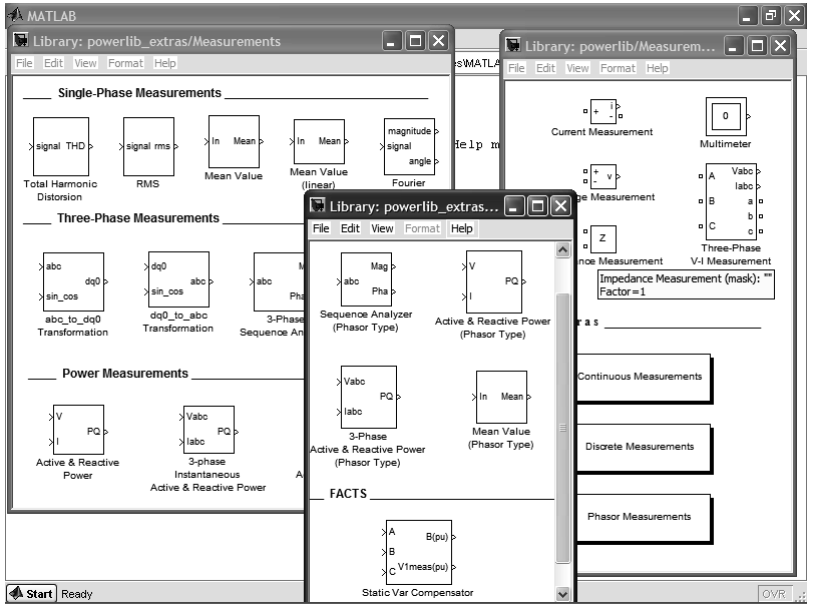


Рис. 11.91. Часть окон библиотеки инструментов Measurement

- **Continuous Measurements** – инструменты непрерывных измерений;
- **Discrete Measurements** – инструменты дискретных измерений;
- **Phasor Measurements** – инструменты фазовых измерений.

Окна первой и третьей групп инструментов показаны на рис. 11.91. Назначение блоков в них вполне очевидно, если учесть определения электрических величин, данных в начале этого урока. Особо следует отметить средства измерения активной P и реактивной Q мощностей в цепях переменного тока. Вычисления выполняются по следующим формулам:

$$P = \frac{1}{T}\int_{(t-T)}^{t}(V(\omega t)\cdot I(\omega t))dt,$$

$$Q = \frac{1}{T}\int_{(t-T)}^{t}(V(\omega t)\cdot I(\omega t-\pi/2))dt.$$

Имеется также обширный набор блоков дискретных измерений. Окно с ними представлено на рис. 11.92. Здесь даны блоки однофазных и трехфазных измерений, блоки измерения мощности и один дополнительный блок (он не виден) в разделе разных инструментов.

Ранее уже неоднократно приводилось применение блоков этой библиотеки. Все они довольно просты, и назначение блоков очевидно из их пиктограмм. При необходимости его нетрудно уточнить по справке. Блоки имеют окно установки

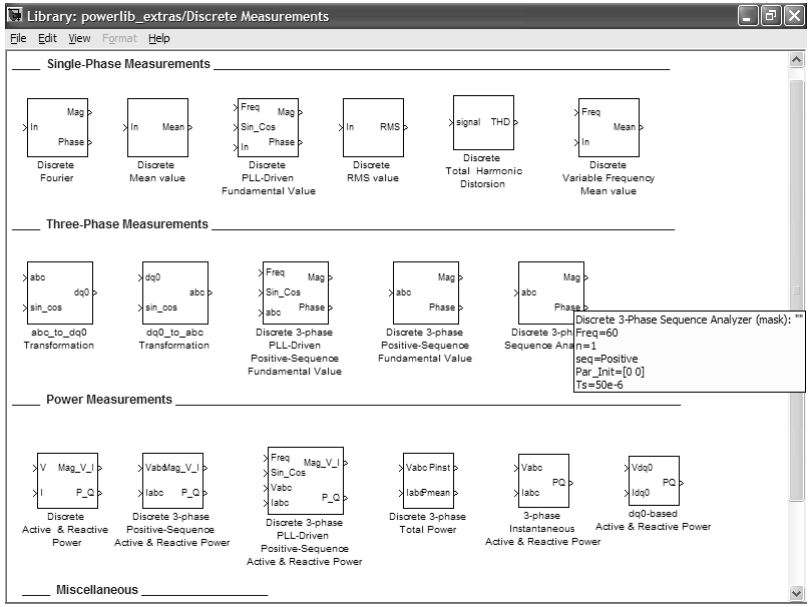


Рис. 11.92. Окно с блоками дискретных измерений

параметров, которое открывается при двойном щелчке мыши на их пиктограмме. Кроме того, если навести курсор мыши на пиктограмму, открывается сноска, на желтом фоне в которой описаны параметры блоков. Так что применение блоков этой группы вполне естественно и в особых пояснениях не нуждается.

11.8.2. Состав библиотеки Extra Library

В состав расширения библиотеки `Extra Library` входят следующие разделы с маскированными блоками:

- `Additional Mashine` – блоки машин постоянного тока `DC Machine` и `DC Discrete Machine`;
- `Control Blocks` – контрольные блоки (6/12-фазные генераторы импульсов, фильтры порядка 1 и 2, программируемый трехфазный источник, триггер, широтно-импульсный модулятор, ждущий генератор импульсов и т. д. – всего 14 блоков);
- `Discrete Measurements` – дискретные блоки измерений (18 блоков однофазных и трехфазных измерений и измерителей мощности);
- `Discrete Control Blocks` – дискретные блоки контроля (25 блоков);
- `Measurement` – блоки измерений (одно- и трехфазных систем и измерения мощности, всего 10 блоков);
- `Phaser Library` – блоки контроля активной и реактивной мощности, блок статического компенсатора и анализатор последовательностей (всего 4 блока);

- `Three-Phase Library` – блоки трехфазных устройств и систем (набор из 18 блоков, поддержка которых прекращается в последующих версиях пакета расширения SimPowerSystem).

Библиотека `Extra Library` содержит много интересных блоков, и пользователю перед построением своих моделей стоит внимательно просмотреть, нет ли уже готовых решений в этой библиотеке. Однако надо помнить, что эта библиотека является все же вспомогательной, и ее блоки представлены в виде маскируемых блоков, которые могут (в отличие от заранее скомпилированных основных блоков библиотек) редактироваться и модифицироваться пользователем. Исполнение таких блоков менее эффективно, чем блоков основных библиотек.

Основное окно этой библиотеки показано на рис. 11.93. Это окно содержит всего пять пиктограмм, соответствующих разделам данной библиотеки. Кроме того, есть пара пиктограмм «obsolete blocks», открывающих разделы библиотеки с устаревшими блоками. Их можно пока применять, но это не рекомендуется, поскольку фирма MathWorks отказывается поддерживать их в будущем. Блоки измерений в данной библиотеке те же, что и дополнительные блоки в библиотеке инструментов Measurements (они описаны выше).

Самое большое число блоков находится в окне раздела дискретных блоков контроля. Оно показано на рис. 11.94. Здесь представлены 17 блоков, назначение которых в основном вполне очевидно.

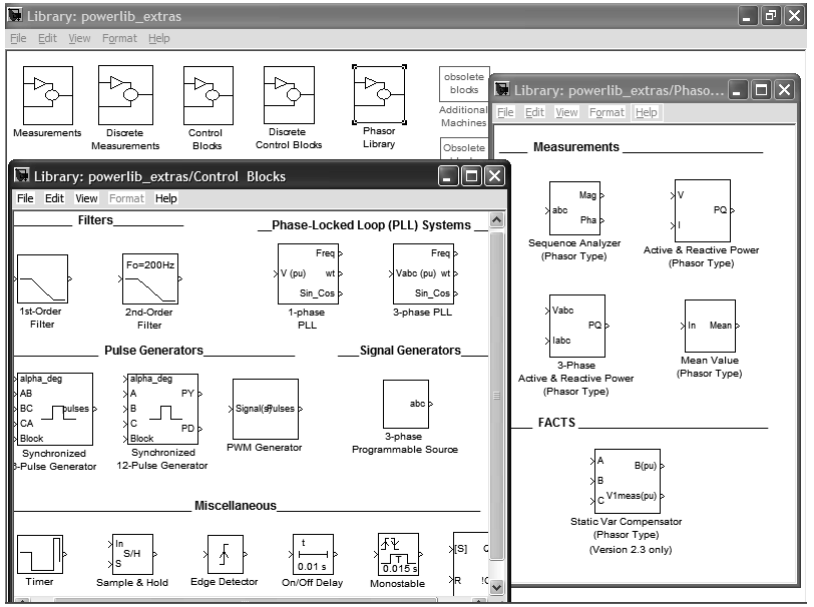


Рис. 11.93. Окно основной библиотеки Extra Library и двух ее подразделов

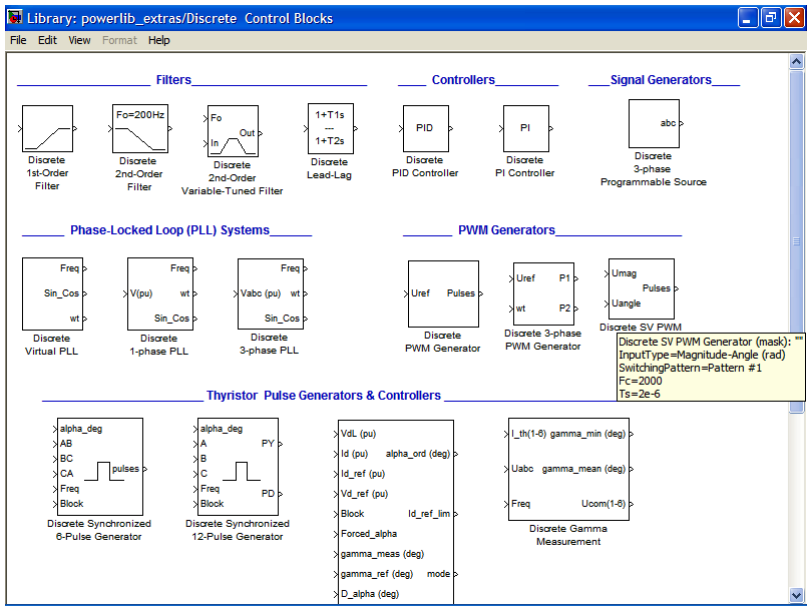


Рис. 11.94. Окно разделов дискретных блоков контроля

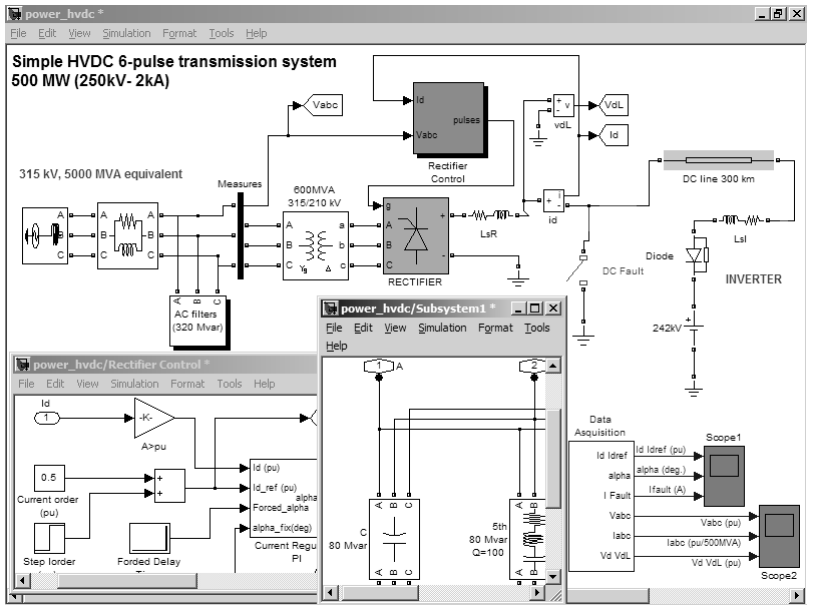


Рис. 11.95. Моделирование сложной энергосистемы большой мощности

11.8.3. Моделирование высоковольтных систем передачи электроэнергии на постоянном токе

Теперь рассмотрим моделирование сложной энергетической системы, показанной на рис. 11.95. Эта система содержит мощный трехфазный источник переменного напряжения (315 кВ, 5000 МВА), мощный реактивный фильтр, 6-импульсный тиристорный преобразователь переменного напряжения в постоянное напряжение, линию передачи на постоянном токе (250 кВ, 2,5 кА), модули постоянного тока и ряд других узлов. Набор средств соответствует современной мощной электростанции.

В этой системе имеется ряд подсистем. Две из них представлены на рис. 11.95: блок контроля выпрямителя **Rectifier Control** и мощный трехфазный реактивный фильтр. В фильтре использован целый ряд трехфазных компонентов из библиотеки **Extra Library**. Осциллограммы работы системы, полученные виртуальными четырехканальными осциллографами, приведены на рис. 11.96. Они свидетельствуют о сложности переходных процессов в этой энергетической системе.

Таким образом, система рис. 11.95 содержит множество разнохарактерных и взаимодействующих друг с другом устройств. В этих условиях выбор метода решения дифференциальных уравнений представляет заметные трудности.

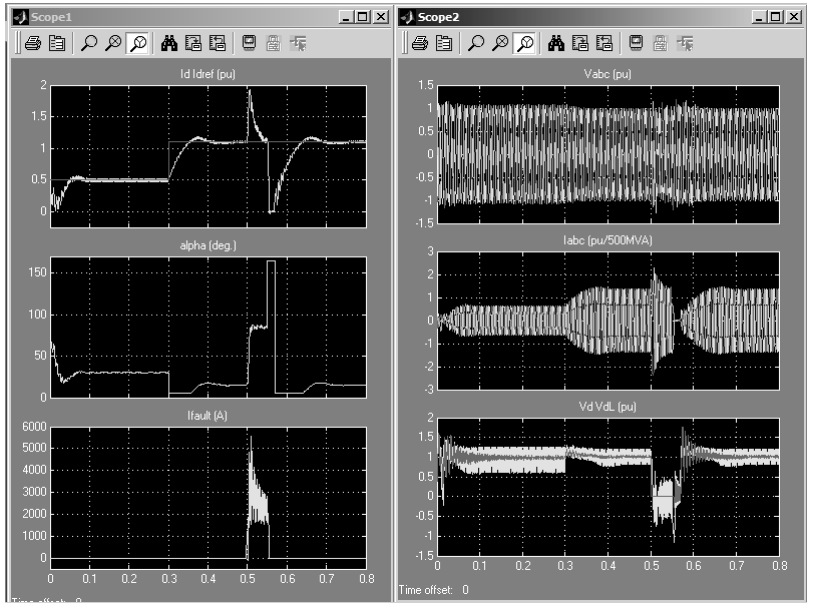


Рис. 11.96. Результаты моделирования энергосистемы большой мощности рис. 11.95

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.

Читатель может самостоятельно ознакомиться с деталями моделирования этих и других демонстрационных примеров применения пакета SimPowerSystems. Но уже из приведенных данных очевидно, что данный пакет может найти самое широкое применение для моделирование электро- и радиотехнических устройств и прежде всего устройств энергетической (силовой) электроники.

На рис. 11.97 показана блок-схема моделирования модной в наши дни мини-энергосистемы ветровой электростанции с асинхронным генератором мощностью 275 кВт с напряжением 480 В и с изолированной сетью.

Здесь особый интерес вызывает организация блока **Continuous (powergui)**. Его пиктограмма представлена на рис. 11.97 в левом верхнем углу. Активизация этой пиктограммы выводит внушительного вида окно **GUI**, показанное на рис. 11.98. Собственно окно **powergui** изображено слева и дает представление о таком окне новейшей версии SimPowerSyatems 4. Другое окно показывает средства проведения спектрального анализа на основе быстрого преобразования Фурье.

Читателю настоятельно рекомендуется познакомиться с другими примерами моделирования энергетических систем и устройств, которые входят в раздел **Demos** справки системы MATLAB + Simulink.

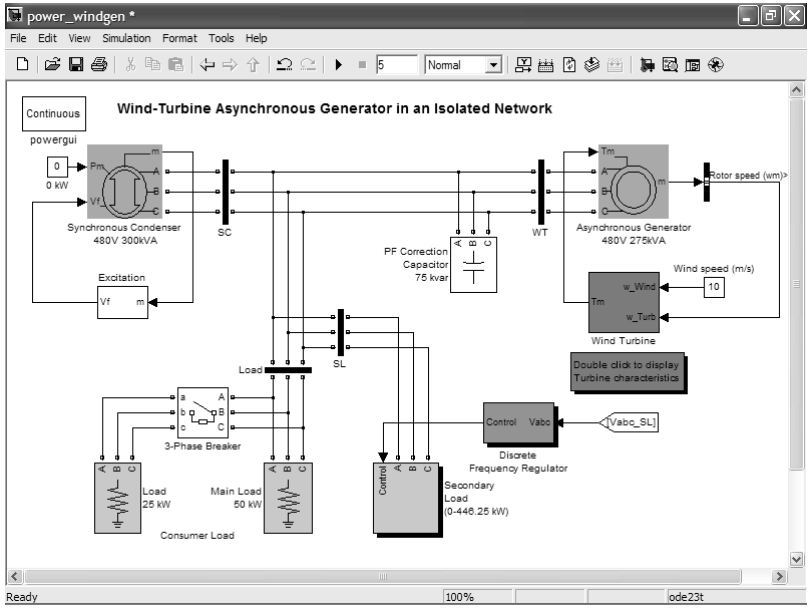


Рис. 11.97. Блок-диаграмма моделирования энергосистемы ветровой электростанции с асинхронным генератором

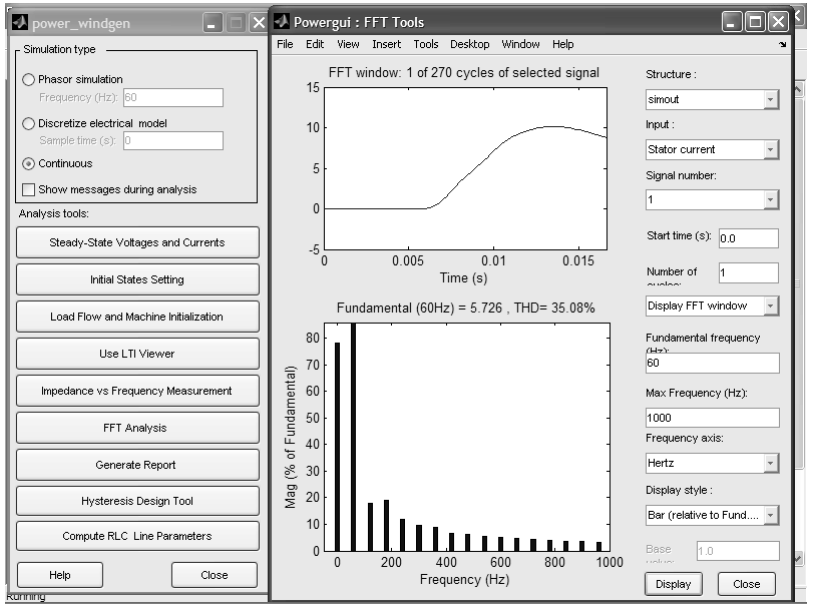


*Рис. 11.98. Окно **powergui** и окно быстрого преобразования Фурье для модели, представленной на рис. 11.97*

Урок 12

Моделирование механических систем и устройств

Механические системы и устройства – еще один класс объектов моделирования, требующих специальных блоков с особыми свойствами. Такие блоки входят в состав пакета расширения SimMechanics. Данный урок посвящен изучению и применению этого интересного и полезного пакета расширения Simulink.

12.1. Начало работы с пакетом SimMechanics Blockset

12.1.1. Назначение пакета SimMechanics и его особенности

Появившийся в новых версиях MATLAB + Simulink пакет расширения SimMechanics Blockset является расширением системы имитационного моделирования Simulink, ориентированным на осуществление математического моделирования механических систем, механизмов и устройств. В пакете широко используется аналогия между моделями механических, электронных и электротехнических систем и устройств. Однако пакет SimMechanics имеет целый ряд весьма специфических особенностей, без учета которых его успешное применение просто невозможно. Им в данном уроке уделено основное внимание.

Пакет SimMechanics развивается сравнительно медленно. В этом уроке описана его версия 2.3, вошедшая в систему MATLAB 7.1 SP 3 + Simulink 6.3. Ее описанные ниже возможности мало отличаются от возможностей предшествующих версий пакета SimMechanics 2.*. В связи с этим приведенные в данном уроке сведения могут использовать и пользователи более ранними версиями системы MATLAB + Simulink.

Прежде всего следует отметить, что пакет оперирует не с сигналами, а с механическими усилиями. Блоки пакета являются моделями механических устройств, положение которых в пространстве и относительно друг друга может меняться в соответствии с законами механики и в разных системах координат. Входы блоков задают «посадочные места» соответствующих механизмов. В силу третьего закона Ньютона «сила действия равна противодействию», связи между «входами» и «выходами» нельзя рассматривать как однонаправленные. Эти связи служат для передачи силовых воздействий, которыми обмениваются части механизма или механизмы между собой. В связи с этим обозначения входов и выходов в SimMechanics особые – не в виде стрелок, а в виде квадратов с диагоналями.

Естественно, что в силу этих особенностей в состав моделей чаще всего нельзя включать блоки из других пакетов расширения, которые имеют обычные входы и выходы. Другими словами, обычные блоки на основе S-моделей, а также их другие средства (окна параметров, вывода информации и т. д.) могут быть неприменимыми. Для устранения этого серьезного ограничения в состав библиотеки пакета SimMechanics включены специальные согласующие блоки.

Моделирование механических систем и устройств осуществляется на основе законов геометрии, физики и механики. Основной целью моделирования является выявление характера движения различных частей механизмов и машин (как в плоскости, так и в пространстве) относительно друг в друга в той или иной системе координат. Разумеется, что при этом учитываются связи между отдельными объектами и различные их движения в соответствии с теми или иными ограничениями.

Пакет SimMechanics поддерживает средства анимации для демонстрации работы механизмов в динамике. Анимация строится на основе средств Microsoft Audio Video Interleave® (AVI), и ее видеофайлы имеют расширение .avi. Поддерживаются и средства OpenGL, в частности при рендеринге (функциональной окраске) трехмерных объектов и поверхностей. Разумеется, что такие виртуальные измерительные средства, как осциллограф и графопостроитель, поддерживаются в моделях (диаграммах) этого пакета.

12.1.2. Библиотека пакета SimMechanics

Как и другие пакеты расширения инструментального ящика **Blockset**, пакет расширения **SimMechanics** имеет свою библиотеку. Доступ к ней из окна браузера библиотек Simulink показан на рис. 12.1.

*Рис. 12.1. Доступ к библиотеке пакета расширения **SimMechanics***

Нетрудно заметить, что разделы библиотеки представлены наглядными для механиков пиктограммами. Полный состав библиотеки виден в окнах, показанных на рис. 12.2 и 12.3. Позже мы обсудим состав блоков, отметив пока наиболее важные из них.

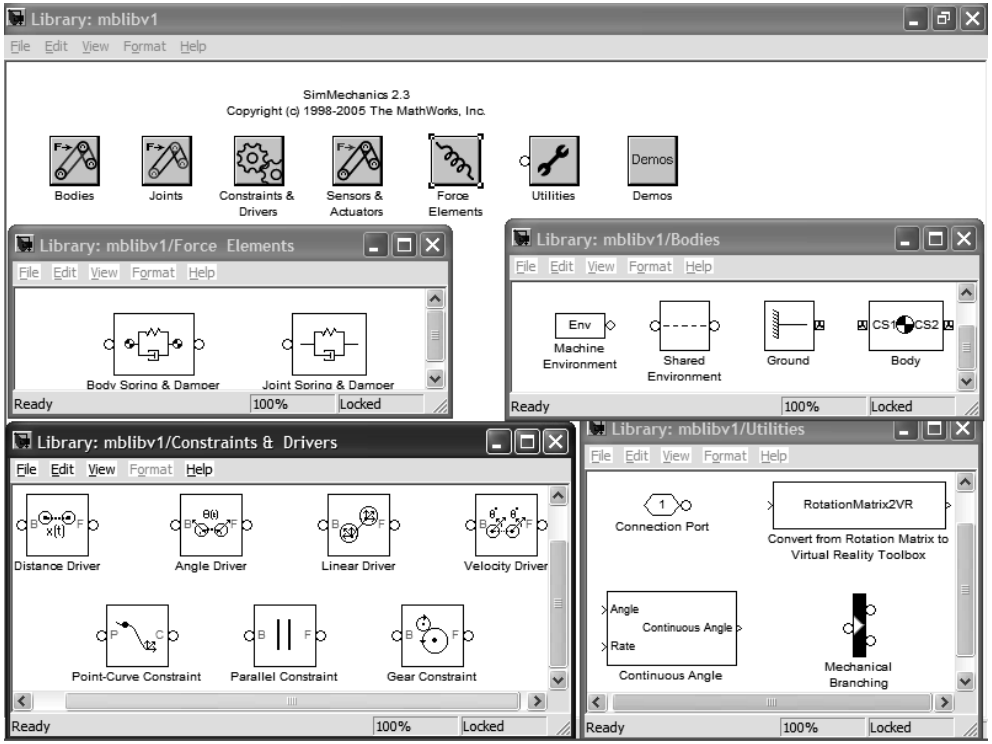


*Рис. 12.2. Окна разделов библиотеки пакета расширения **SimMechanics***

Библиотека пакета **SimMechanics Blockset** содержит следующие разделы:

- **Bodies** – четыре блока (основные: корпус и основа – земля);
- **Constrains & Drives** – три устройства принуждения (ограничения) и четыре драйвера (расстояния, угла, скорости и линейный);
- **Force Elements** – силовые устройства и амортизаторы (2 блока);
- **Joints** – соединители и сочленители (17 блоков);
- **Sensors & Actuators** – сенсоры, силовые приводы и блоки стыковки с блоками Simulink (9 блоков);
- **Utilites** – утилиты (4 блока);
- **Demos** – демонстрационные примеры.

Как следует из представленного описания, библиотека пакета SimMechanics Blockset сравнительно невелика. Однако надо учитывать, что для подготовки математических моделей механических систем и устройств можно (и нужно!) использовать некоторые блоки пакета расширения Simulnk, например блоки регистраторов, математических и логических операций и т. д. С учетом этого можно строить модели множества механических систем и устройств. Раздел библиотеки **Demos** дает доступ к множеству демонстрационных примеров, хорошо выявляющих особенности применения пакета **SimMechanics** в моделировании механических устройств.

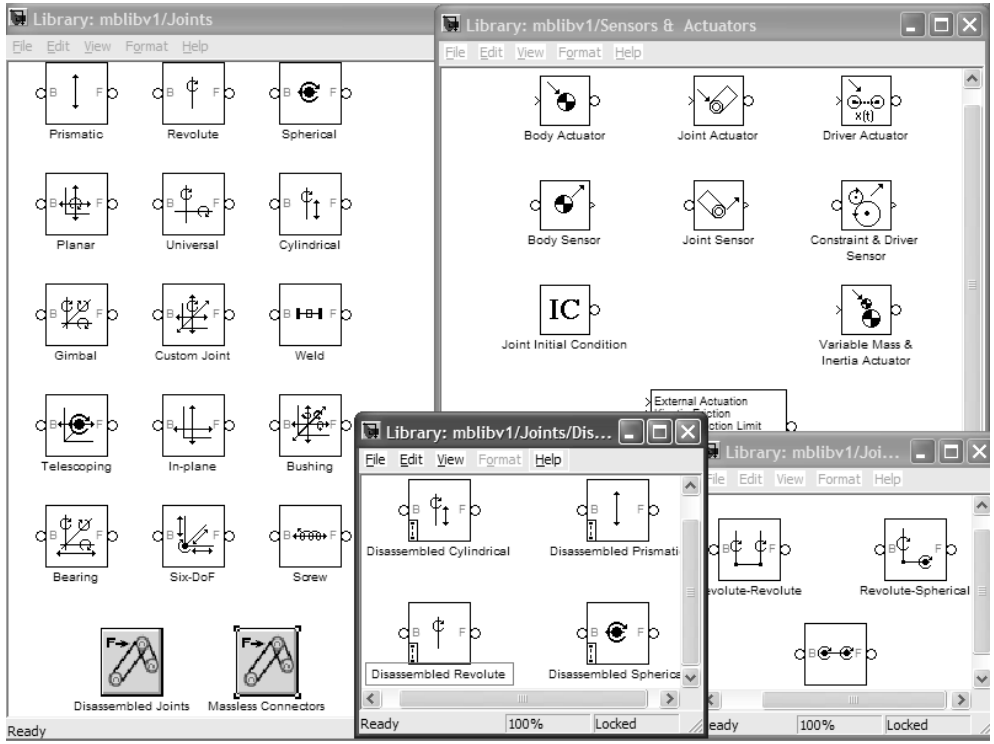


*Рис. 12.3. Окна разделов библиотеки пакета расширения **SimMechanics***

Раздел **Demos** основного окна библиотеки самостоятельного значения не имеет. Активизация его пиктограммы выводит окно справки с открытой вкладкой **Demos** по пакету **SimMechanics Blockset**. Это окно показано на рис. 12.4. Оно демонстрирует довольно большой набор примеров, по которым можно достаточно детально изучить возможности этого пакета расширения. Объективности ради стоит отметить, что о некоторой недоработке пакета говорит отсутствие индивидуальных графических обозначений примеров применения данного пакета расширения.

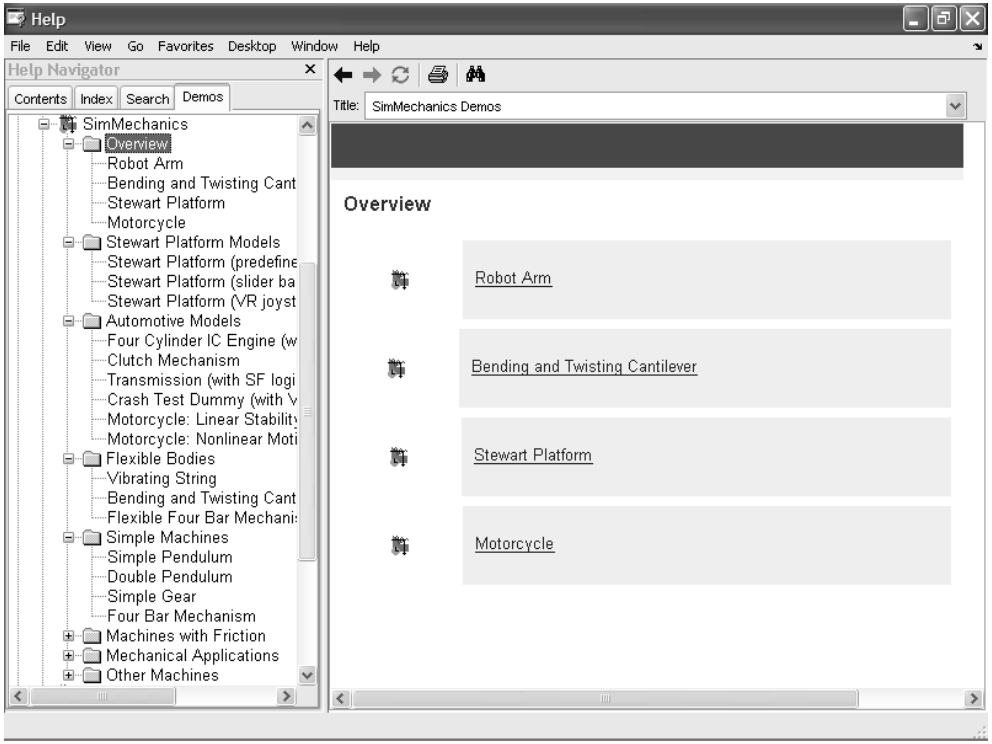


*Рис. 12.4. Окно раздела **Demos** справки пакета расширения **SimMechanics***

12.1.3. Раздел библиотеки Bodies

Раздел библиотеки **Bodies (Тела)** содержит всего четыре блока:

- **Machine Environment** – блок машинной окружающей среды;
- **Shared Environment** – блок разделенной среды;
- **Ground** – блок задания основания;
- **Body** – блок задания тела.

В старых версиях пакета два первых блока отсутствовали. Они служат для установки параметров сред модели, и их окна содержат ряд важных установок, которые мы рассмотрим несколько позже. Блок **Ground** служит для задания неподвижного основания со своей координатной системой, в которой происходит движение тела, заданного блоком **Body**.

12.1.4. Системы координат SimMechanics

Перемещение тех или иных деталей механизмов происходит в пространстве. Поэтому важно установить соглашения относительно координатных систем в пакете SimMechanics. Прежде всего надо отметить мировую систему координат World,

центр которой обозначается как (0, 0, 0). Однако отдельные детали механизмов часто передвигаются в своих координатных системах блоков **Ground**, центры которых смещены относительно начала координат мировой координатной системы.

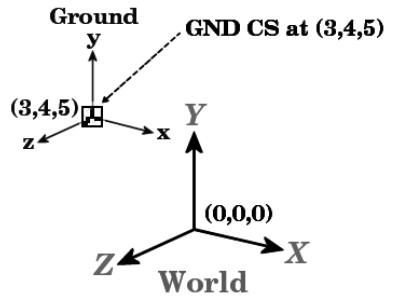


*Рис. 12.5. Координатная система блока **Ground** в инерциальной системе координат World*

Пиктограмма блока **Ground** показана в окне **Bodies** на рис. 12.2. Расположение координатной системы блока **Ground** в мировой системе координат представлено на рис. 12.5. Координаты центра координатной системы основания Ground (в нашем примере это (3,4,5)), указанные в инерциальной (мировой) системе координат с центром (0,0,0), являются единственным параметром блока **Ground**.

Из сказанного вытекает, что любая диаграмма пакета SimMechanics должна содержать хотя бы один блок **Ground**. Этот блок характеризуется координатами центра его координатной системы в мировой системе координат. Прочие блоки мы рассмотрим по мере описания примеров применения пакета расширения SimMechanics.

12.2. Простой пример моделирования механического маятника

12.2.1. Диаграмма простого механизма – стержневого маятника

Как и при моделировании других устройств, в SimMechanics оно начинается с подготовки диаграммы модели путем переноса мышью нужных блоков из разделов библиотеки SimMechanics в окно диаграммы и соединения их нужными связями. Этот процесс ничем не отличается от ранее описанного для диаграмм расширения Simulink. Можно загрузить и готовую диаграмму, например из обширного перечня демонстрационных примеров в разделе **Demos** для этого пакета. Начнем знакомство с диаграммой с простого примера Simple Pendulum (Простой маятник) на моделирование поведения подвешенного одним концом на вращающемся валу стержня.

Стержень, закрепленный на вращающемся валу, при наличии сухого трения может совершать периодические незатухающие колебания. Простая диаграмма, показанная на рис. 12.6, демонстрирует модель такого маятника.

Здесь тело (стержень) Body соединено с валом (блок **Revolute**). Фиксация устройства в пространстве обеспечивает блок основания **Ground**. При вращении вала против часовой стрелки стержень поворачивается в этом направлении под

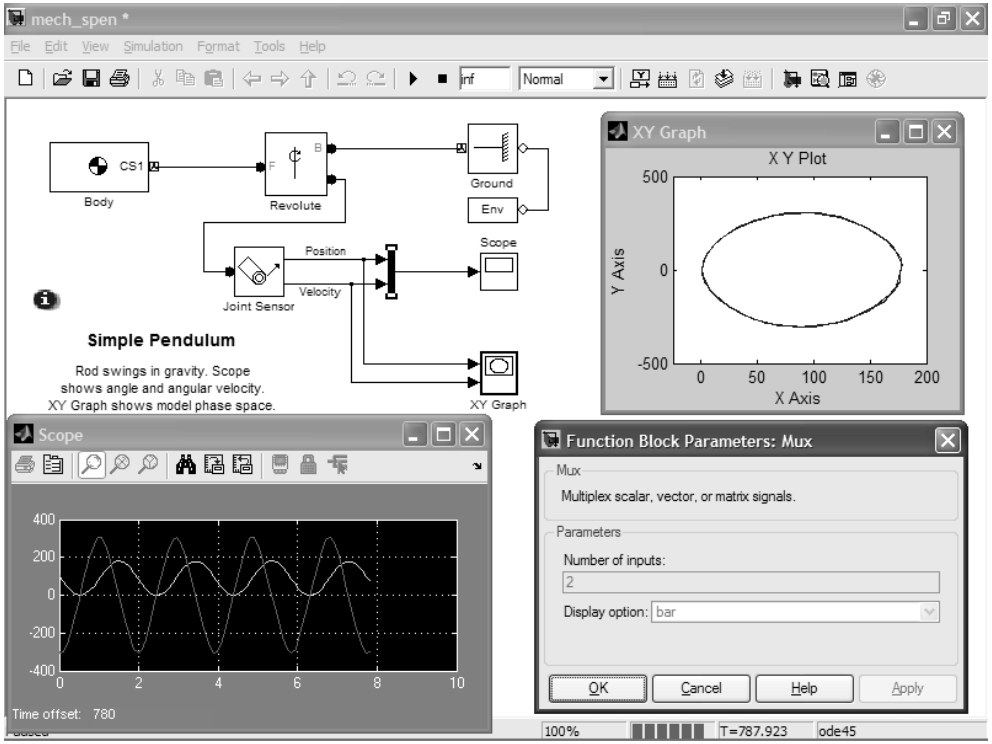


Рис. 12.6. Диаграмма простого механизма – стержневого маятника

действием силы трения. Но в какой-то момент он срывается и начинает движение по часовой стрелке. Дойдя на крайнего левого положения, он начинает вновь поворачиваться в направлении по часовой стрелке и т. д. Таким образом, возникают периодические колебания вала вокруг точки его подвески, и данная система моделирует механический маятник с незатухающими колебаниями.

12.2.2. Пуск модели и наблюдение результатов моделирования

Пуск модели самый обычный – активизацией кнопки с зеленым треугольником. Поскольку в демонстрационном примере параметры блоков заданы по умолчанию, можно выполнить пуск сразу после загрузки диаграммы модели командой **Open** в позиции меню **File** окна диаграмм (или библиотеки) Simulink.

Для контроля колебаний стержня в построенной диаграмме служат графопостроитель и виртуальный осциллограф. Для преобразования положения конца колеблющегося стержня и скорости его колебаний в сигналы, воспринимаемые осциллографом и графопостроителем, в модель включен блок **Joint Sensor**. Графопостроитель строит фазовый портрет колебаний, а осциллограф – их временные зависимости. Колебания в нашей механической системе близки к гармоническим (синусоидальным), а фазовый портрет их очень близок к эллипсу. Окна указанных регистрирующих устройств показаны на рис. 12.6. Там же представлено окно установки параметров блока **Mux**.

12.2.3. Специальные средства визуализации пакета SimMechanics

В пакете SimMechanics есть довольно развитые средства для более наглядной визуализации движения механических устройств. Чтобы ввести их в действие, надо вывести окно параметров конфигурации, показанное на рис. 12.7. Для вызова этого окна достаточно исполнить команду **Configuration Parameters...** в позиции **Simulation** меню окна диаграмм Simulink. Окно рис. 12.7 показано с открытой вкладкой **SimMechanics**, на которой надо отметить птичкой опции визуализации в разделе **Visualization** (по умолчанию они отключены).

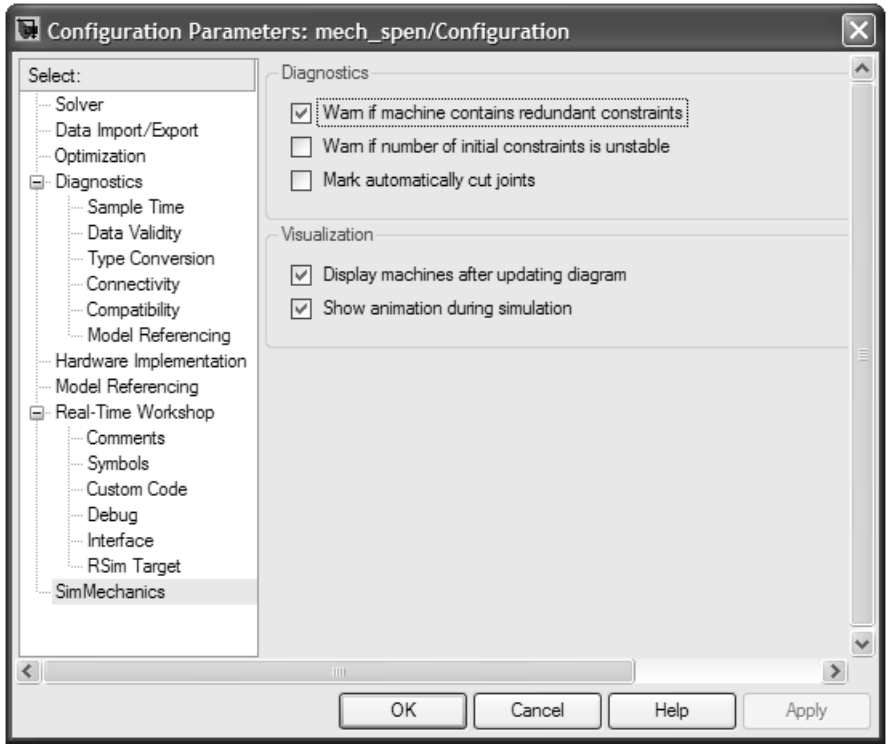


*Рис. 12.7. Окно **Configuration Parameters** с открытой вкладкой **SimMechanics***

Теперь при запуске диаграммы можно увидеть появление окна визуализации, показанного на рис. 12.8. Внизу окна будет виден колеблющийся стержень, а сверху – ряд панелей инструментов окна и его обычное меню. Это окно является стандартным и применяется для визуализации работы в динамике и других механизмов.

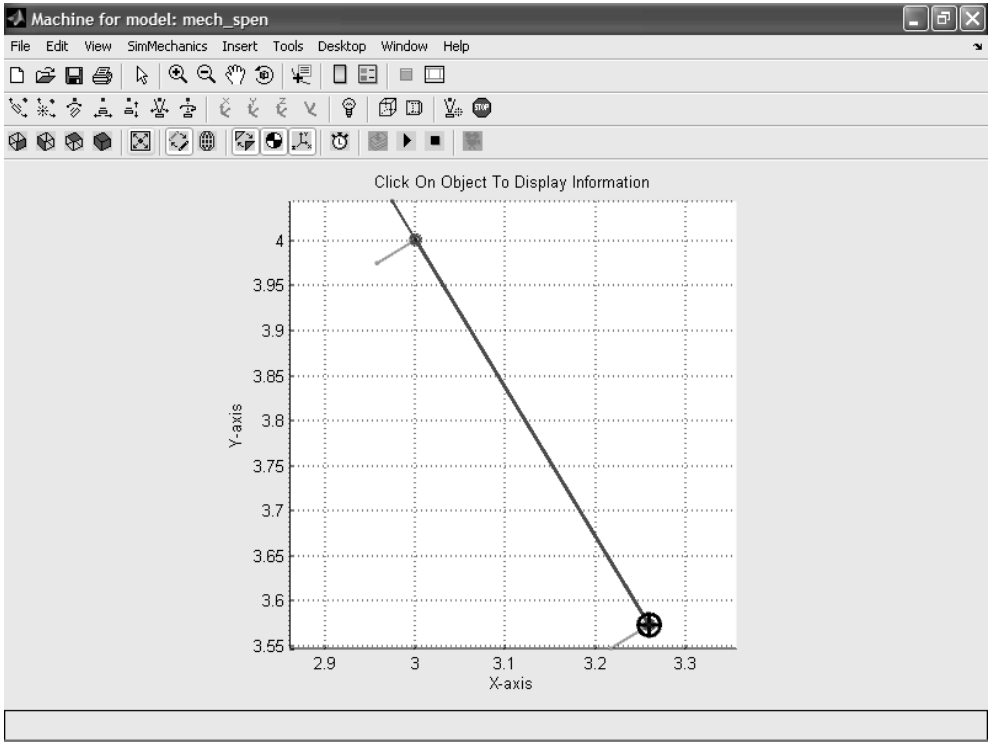


Рис. 12.8. Окно визуализации колебаний простого маятника

Объем этого небольшого урока не позволяет детально описать все возможности настройки визуализации результатов моделирования механических устройств. Но в этом и нет особой необходимости – обозначения на кнопках панелей инструментов достаточно красноречивы, а заинтересованный пользователь этим пакетом может за несколько минут экспериментов с окном понять, что позволяют делать эти кнопки. Например, с их помощью можно выводить шкалу цветов, список обозначений (легенду), менять вид стержня маятника, включать или выключать отдельные элементы изображения и т. д. и т. п. На рис. 12.9 показано окно визуализации колебаний маятника после подобных операций форматирования изображения.

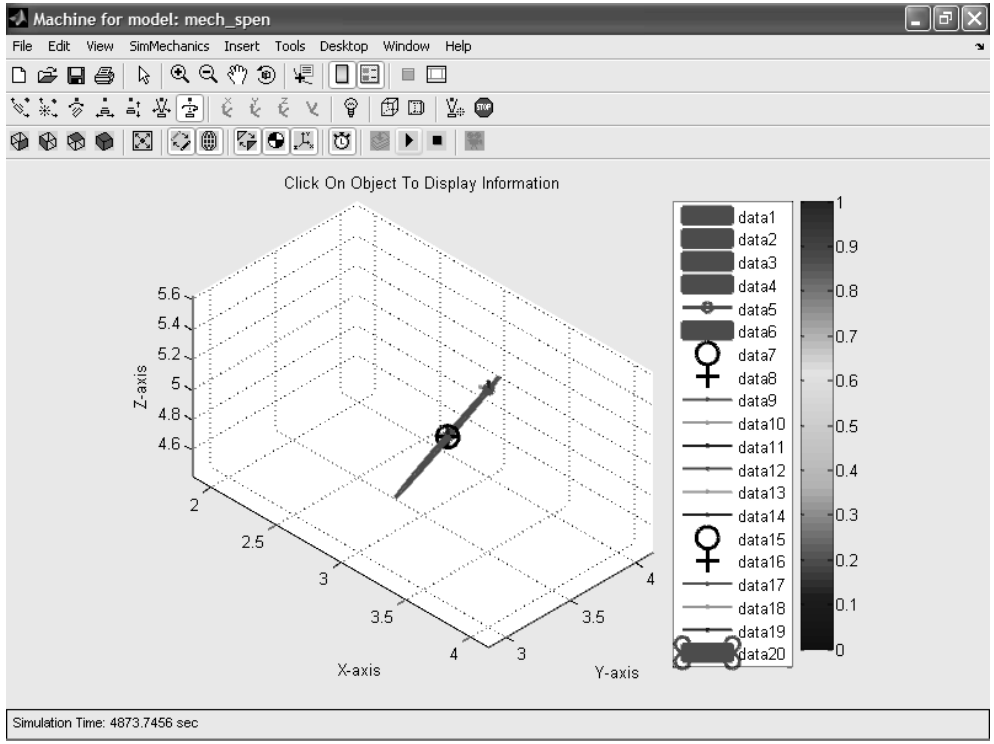


Рис. 12.9. Окно визуализации колебаний маятника после форматирования изображения

12.2.4. Установка параметров блоков диаграммы

Для правильного функционирования любой модели (диаграммы) необходима установка соответствующих параметров блоков. На рис. 12.6 представлено окно одного из блоков – **Mux**. Этот блок объединяет сигналы положения и скорости в один сигнал для подачи на вход виртуального осциллографа. Он имеет всего два параметра – число входов и вид обозначения. Большинство блоков пакета **SimMechanics** имеет гораздо большее число параметров, и некоторые из них достаточно специфические.

Здесь мы вначале рассмотрим окна установки параметров раздела библиотеки **Bodies**. Для блока **Ground**, как отмечалось, устанавливается только один параметр – вектор координат центра новой системы координат относительно координат центра инерциальной систем координат.

На рис. 12.10 показано окно самого важного блока этого раздела **Body**. Блок имитирует перемещаемое в пространстве тело, которое является частью моделируемого механизма.

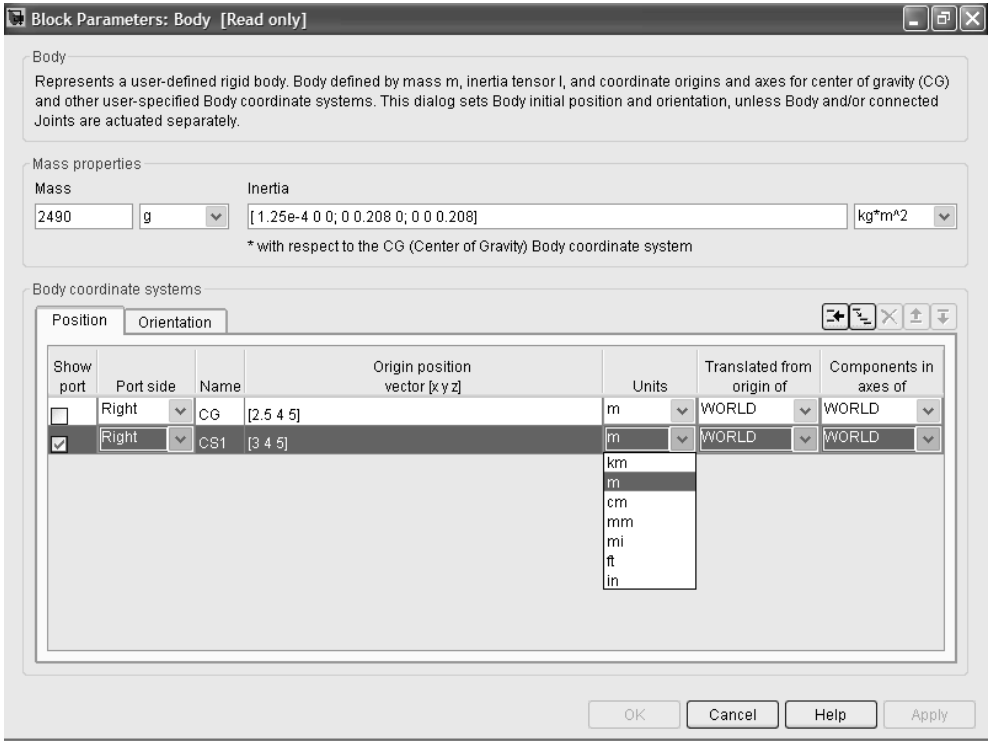


*Рис. 12.10. Окно установки параметров блока **Body** с открытой вкладкой **Position***

Окно установки параметров блока **Body** содержит два раздела. Первый раздел **Mass Properties** задает свойства масс. В этом разделе устанавливаются масса тела в той или иной системе единиц (выбор из списка) и момент инерции тела (также с выбором из списка нужной единицы измерения). Момент инерции задается вектором или матрицей (в нашем примере механизма на плоскости это вектор из трех компонент). В общем случае момент инерции задается матрицей размера 3×3, задающей моменты инерции относительно ортогональных осей, жестко связанных с телом и проходящих через центр тяжести тела.

В нижней части окна имеются две вкладки – параметров положения (позиции) **Position** и ориентации **Orientation**. На рис. 12.10 окно показано с открытой вкладкой **Position**. На этой вкладке устанавливаются параметры позиции тела в нужных системах координат. Обязательно должна присутствовать система координат центра тяжести тела CG (Center of Gravity – центр тяжести тела), относительно осей которой и задаются моменты инерции тела.

На вкладке **Position** задаются следующие параметры:

- **Show Port** – показ (при установке флажка) или скрытие изображения порта;
- **Port Side** – задание изображения порта слева или справа в изображении блока;

- **Name** – имя системы координат (выбор из списка);
- **Origin position vector [x,y,z]** – вектор координат начала заданной системы координат;
- **Units** – выбор из списка единицы измерения (на рис. 12.10 список показан в открытом состоянии и демонстрирует набор единиц измерения);
- **Translated from origin of** – отсчет относительно начала выбранной из списка системы координат;
- **Component in axes of** (компоненты в осях системы координат) – задание из списка имени системы координат, относительно которой отсчитываются координаты установленной для компонента системы координат.

Другая вкладка **Orientation** представлена на рис. 12.11. На ней устанавливаются следующие параметры:

- **Show Port** – показ (при установке флажка) или скрытие изображения порта;
- **Port Side** – задание изображения порта слева или справа в изображении блока;
- **Name** – имя системы координат (выбор из списка);
- **Orientation vector [x,y,z]** – вектор углов поворота выбранной системы координат относительно исходной системы координат;

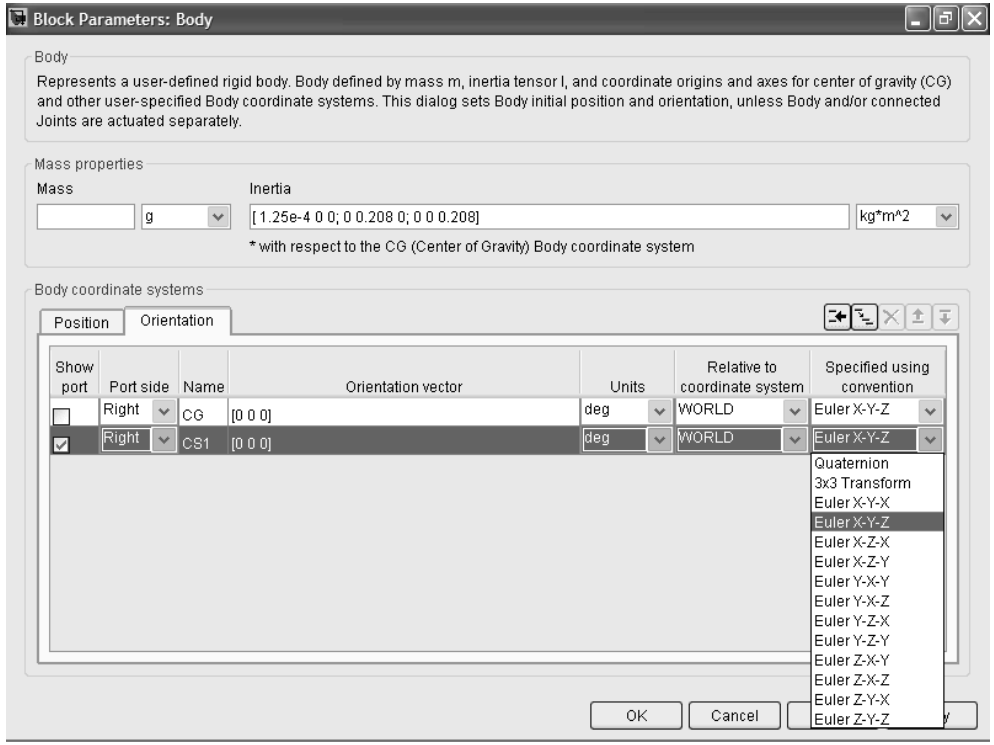


*Рис. 12.11. Окно установки параметров блока **Body** с открытой вкладкой **Orientation***

- **Units** – выбор из списка единицы измерения;
- **Related to coordinate system** – отсчет относительно начала выбранной из списка системы координат;
- **Specified using convention** – задание спецификации исходной системы координат с выбором из списка (он показан на рис. 12.11 в открытом виде).

На рис. 12.12 показаны окна установки параметров еще двух блоков. Окно параметров блока **Revolute** содержит данные о координатных системах и числе портов, которое может меняться. На главной вкладке **Axes** (**Оси**) задаются имя и тип примитива, а также указывается (значением 1) ось, относительно которой осуществляется вращение.

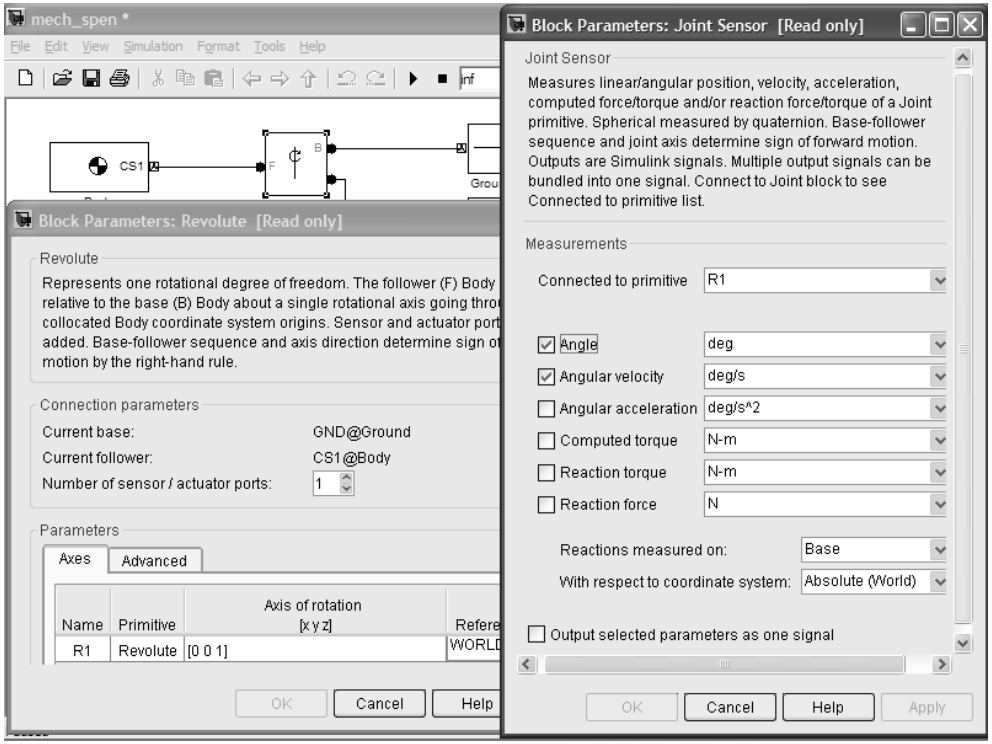


*Рис. 12.12. Окна параметров блоков **Revolute** и **Joint Sensor***

Блок **Joint Sensor** имеет достаточно большое число параметров – опций. Вначале надо задать, к какому примитиву относится подключение, а затем по мере необходимости ввести следующие опции:

- **Angle** – угол поворота выхода примитива Follower относительно входа Base;
- **Angular velocity** – угловая скорость (относительная);
- **Angular acceleration** – угловое ускорение (относительное);

- **Computed torque** – вычисленный полный момент сил, вызывающий угловое ускорение;
- **Reaction torque** – момент реакции относительно оси примитива;
- **Reaction force** – сила реакции по оси примитива.

Для каждого параметра из списка нужно выбрать нужную единицу измерения. Помимо задания указанных опций, можно задать дополнительные опции: где измеряется реакция и в какой координатной системе, нужно ли вывести измеряемые параметры как один сигнал.

12.3. Идеологии пакета SimMechanics

12.3.1. Наглядное представление механических устройств

В основе идеологии пакета расширения **SimMechanics** лежит максимальное приближение его изобразительных средств к тем, что используются в механике. Для понимания рассмотренных ниже примеров применения пакета **SimMechanics** рассмотрим гипотетический конвейерный механизм, описанный в справке по пакету и приведенный на рис. 12.13. На этом рисунке приведены также обозначения различных частей данного механизма, характерные для массы других механизмов.

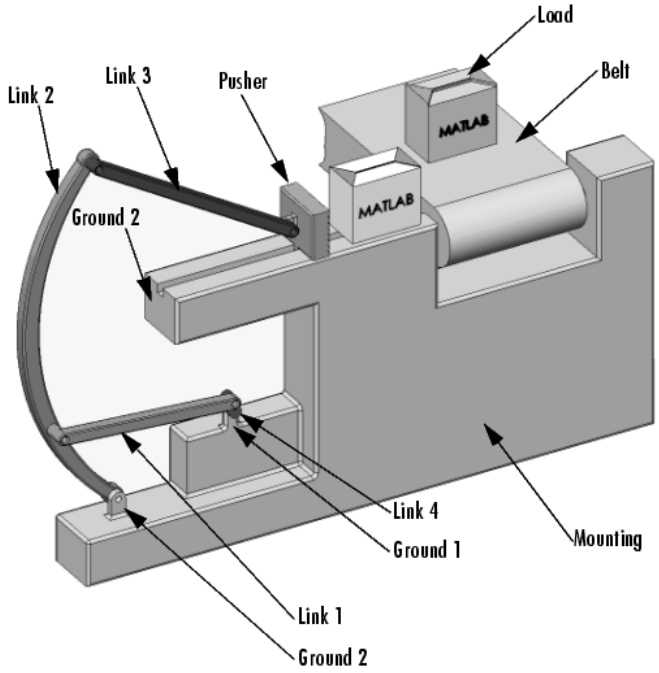


Рис. 12.13. Гипотетический механизм

Рисунок 12.13 поясняет принципы визуализации механизмов, положенные в основу пакета расширения **SimMechanics**, – они такие же, как при обычном аксонометрическом изображении трехмерных объектов. Разумеется, возможно и более простое изображение механизмов, например в виде плоских объектов. В любом случае визуализация механизмов осуществляется в стиле механики, в чем мы еще не раз убедимся и на других примерах.

Рассмотрим назначение основных частей механизма рис. 12.13:

- **Belt** – ремень ременной передачи;
- **Ground** – основание или крепление (аналог общего провода или земли в блоках электро- и радиотехнических устройств). Оснований может быть несколько, и они характеризуются своими координатами в абсолютном (инерциальном) пространстве и номерами;
- **Link** – механическая связь;
- **Load** – нагрузка;
- **Mounting** – основа (подложка) механизма;
- **Pusher** – толкатель.

Каждый узел механизма характеризуется рядом физических и геометрических параметров. Геометрические параметры обычно берутся из машиностроительных чертежей механизма. Такой чертеж для механизма рис. 12.13 представлен на рис. 12.14. Чертеж дает представление о реальных размерах механизма и статическом расположении его узлов.

Каждая деталь механизма описывается моделью, содержащей один или несколько блоков. Нередко они образуют цепи **Ground-Joint-Body-Joint-…-Body**, где хотя бы одно из тел должно быть представлено основанием **Ground**. Для имитации механических связей используются блоки **Driver** и **Constraint**.

Блоки диаграмм пакета имеют обычные окна установки параметров (см. примеры выше при описании простого маятника). Для их вызова достаточно дважды щелкнуть мышью на нужном блоке. Для блоков **Body**, **Sensor** и **Driver** в окнах установки параметров можно задать подключение блоков **Actuator** и **Sensor**. При этом блок **Actuator** позволяет задать временную зависимость движения, а блок **Sensor** – вывести результаты моделирования в рабочее пространство системы MATLAB.

12.3.2. Пример диаграммы конвейерного механизма

С учетом описанного выше можно составить диаграмму конвейерного механизма, которая приведена на рис. 12.15 и имеется в файле демонстрационных примеров mech_conveyor. Проследив за схемой соединений отдельных блоков в диаграмме, нетрудно убедиться в том, что она вполне адекватна схеме передачи механических усилий в представленном на рис. 12.13 конвейерном механизме.

Диаграмма, изображенная на рис. 12.15, представляет собой типичную Simulink-модель, которая используется так же, как и другие диаграммы. Например, ее

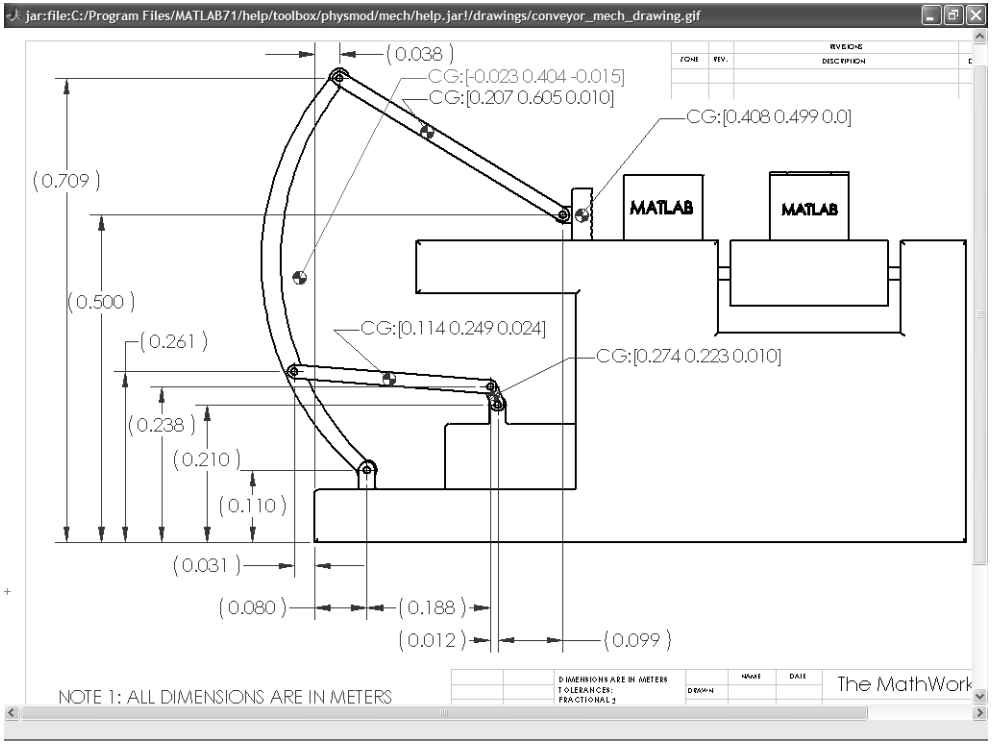


Рис. 12.14. Чертеж механизма, представленного на рис. 12.13

можно редактировать, изменять положение блоков, выводить окна параметров блоков и т. д. Так, на рис. 12.15 справа показано окно параметров блока **Ground3**. Нетрудно заметить, что единственным параметром этого блока является вектор координат центра координатной системы данного блока, записанных относительно центра мировой системы координат.

Блоки диаграмм в пакете **SimMechanics** могут содержать субблоки. На диаграмме рис. 12.15 таким субблоком является контроллер позиции **Position Controller** (выделен серым цветом).

Для визуализации результатов моделирования могут использоваться различные средства. Чаще всего используются виртуальные осциллографы, которые, разумеется, отражают временные зависимости уже механических, а не электрических величин. Окно осциллографа показано на рис. 12.15 под окном параметров блока **Ground3**.

В демонстрационных примерах пакета **SimMechanics** параметры блоков заданы по умолчанию, что позволяет сразу запускать диаграмму и получать характерные результаты моделирования. Разумеется, пользователь может корректировать параметры блоков, а в случае подготовки своих моделей он должен задавать их самостоятельно.

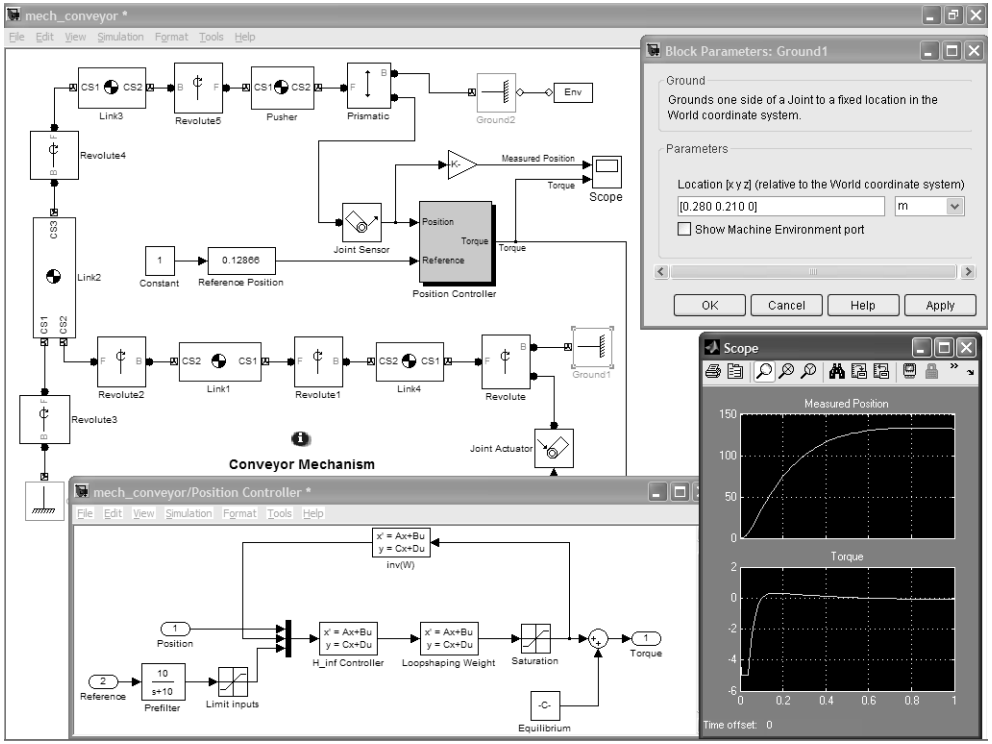


Рис. 12.15. Диаграмма (модель) конвейерного механизма, представленного на рис. 12.13

12.3.3. Контроль общих установок моделирования

Перед пуском диаграммы рекомендуется проверить общие установки моделирования. Для этого в позиции **File** меню окна Simulink достаточно исполнить команду **Preferences** – появится окно, показанное на рис. 12.16, с открытой вкладкой **Solve**. На ней представлены установки, относящиеся к решателям дифференциальных уравнений, на которых базируется моделирование в расширении Simulink. На этой вкладке можно изменить тот или иной параметр моделирования – например, взятого по умолчанию конечного времени, равного бесконечности, задать конечное время (показано его значение, например 1 или 2). Можно выбрать метод моделирования и тип решателя системы дифференциальных уравнений.

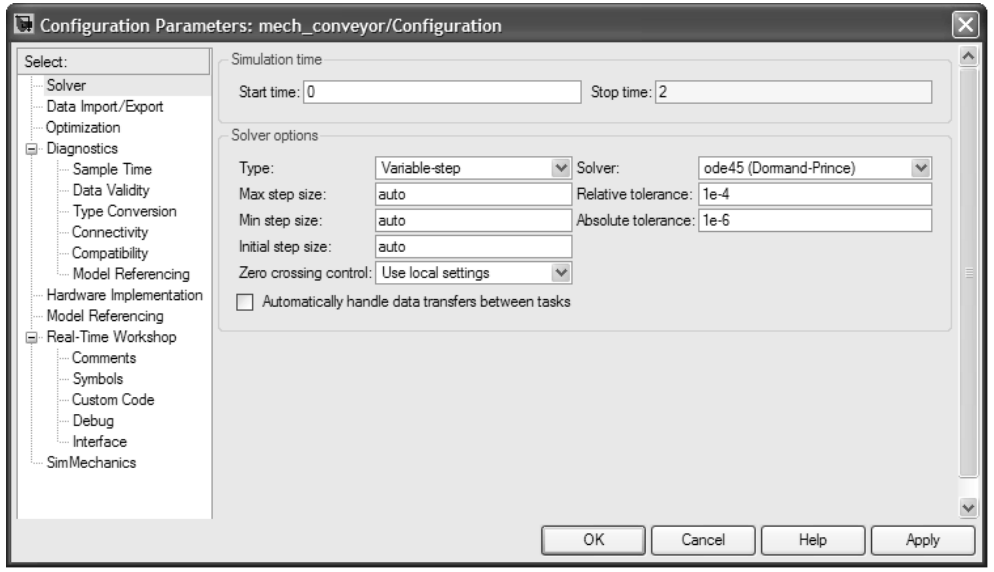


*Рис. 12.16. Окно **Preferences** с открытой вкладкой **Solve***

12.3.4. Пуск модели механизма конвейера

Теперь практически все готово к моделированию описанного выше устройства. Как уже отмечалось, для пуска моделирования достаточно активизировать кнопку пуска (с зеленым треугольником) на панели инструментов окна диаграммы. По завершении моделирования можно просмотреть осциллограммы результатов моделирования, представленные на рис. 12.17. Для автоматического масштабирования осциллограмм активизируйте кнопку панели инструментов окна осциллографа с изображением бинокля. Осциллограммы рис. 12.17 получены после этого и иллюстрируют временные зависимости положения и вращающего момента для блока **Joint Sensor**.

Как уже отмечалось на примере простого маятника в SimMechanics, возможна улучшенная визуализация результатов модели-

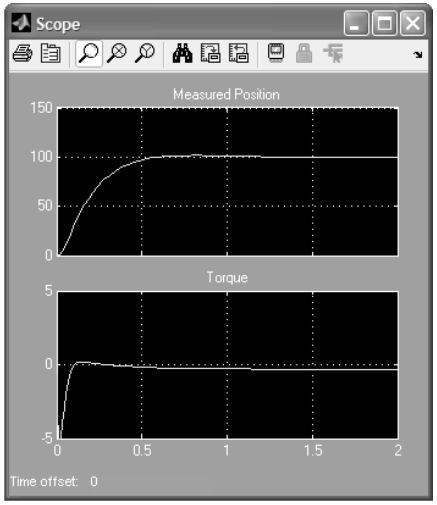


Рис. 12.17. Окно осциллографа с осциллограммами конвейерного механизма

рования работы механизмов. На рис. 12.18 показано окно такой визуализации с представлением механизма в пространстве.

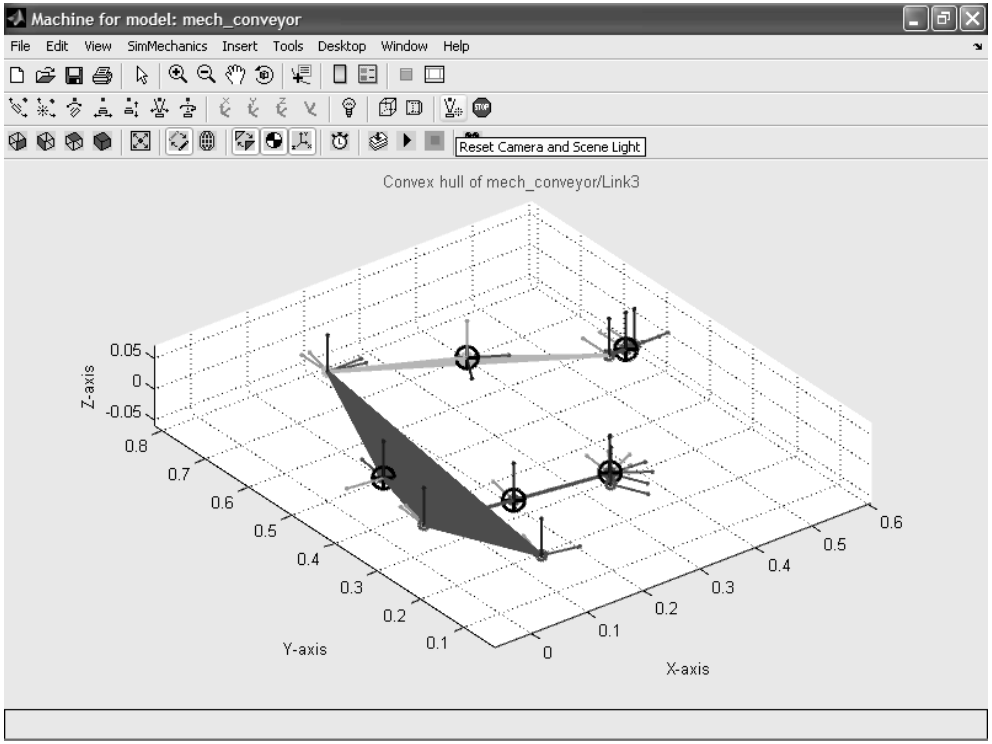


Рис. 12.18. Окно улучшенной визуализации конвейерного механизма

12.4. Обзор основных блоков библиотеки SimMechanics

Выше мы уже рассмотрели блоки раздела **Bodies (Тела)**. Настало время рассмотреть блоки из других разделов библиотеки пакета расширения SimMechanics.

12.4.1. Блоки раздела Joints (Сочленения)

Блоки раздела **Joints (Сочленения)** служат для организации различного рода сочленений между отдельными частями (деталями) механизма, представленными блоками раздела **Bodies**. Для иллюстрации организации сочленения рассмотрим простую диаграмму механизма, показанную на рис. 12.19 в окне **untitled*** (оно изображено справа). Эта диаграмма строится переносом трех блоков **Ground**,

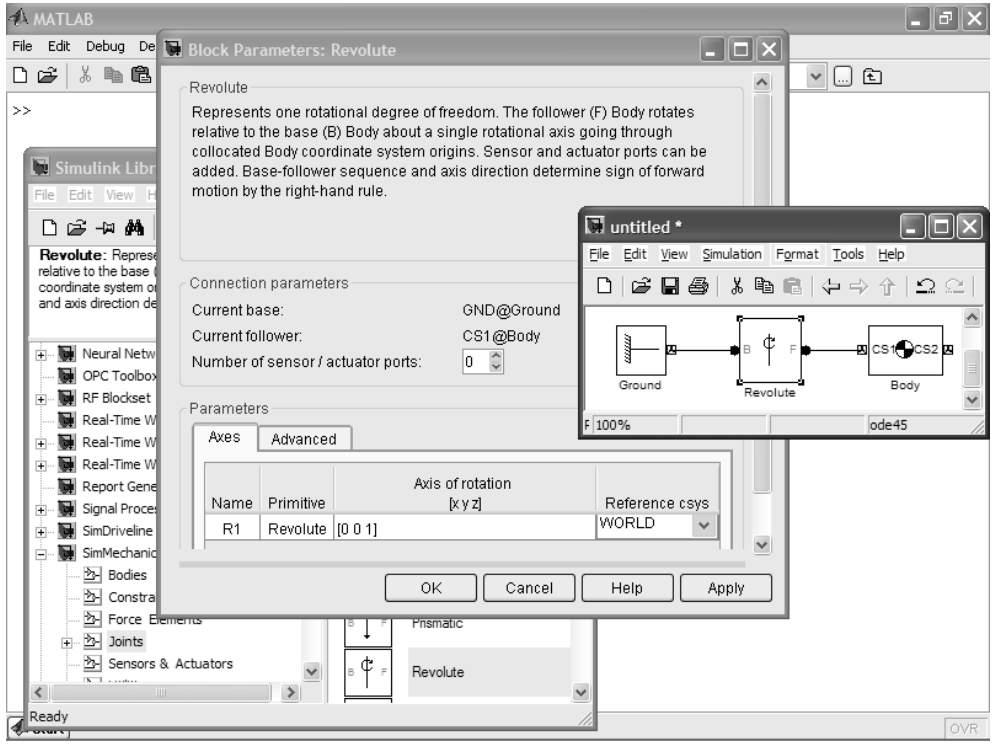


Рис. 12.19. Пример построения простейшего механизма из трех блоков

Resolve и **Body** с окна библиотеки **SimMechsnics** (показано снизу слева), соединенных обычными соединениями.

Изначально блоки раздела **Joints** (см. окно с ними на рис. 12.3 слева) имеют два обязательных порта, обозначенных как **B** (Body) и **F** (Follow). К порту **B** подключается основное тело – в нашем случае это основание, а к порту **F** – следующее тело (в нашем случае Body).

Теперь, дважды щелкнув мышью на блоке **Resolve**, можно вывести окно установки параметров этого блока. Оно показано на рис. 12.19 в центре. Отметим более подробно информацию, выводимую в этом окне, и назначение его параметров. Группа параметров подключения (Connections parameters) представлена тремя параметрами:

- **Current base** – информация о подключенном основном теле;
- **Current follower** – информация о подключении следующего тела;
- **Number of sensor / actuator ports** – число дополнительных портов сенсоров/возбудителей.

Первые два параметра чисто информационные. Знак @ указывает на соединение двух объектов. Изначально дополнительные порты отсутствуют, и их число обозначено как 0. Если указать 1, то появится дополнительный порт (см. рис. 12.12,

где в окне параметров установлено число дополнительных портов, равное 1, а на изображении блока **Resolve** виден дополнительный порт). Если блок раздела **Joint** не подсоединен к телам, то напротив параметров появляется надпись «no connected» (не подключен). Назначение дополнительных параметров блока **Resolve** уже описывалось.

Всего в разделе библиотеки **Joints** имеются 15 основных блоков и два дополнительных. Отметим назначение основных блоков:

- **Prismatic** – призматическое сочленение, дающее поступательные движения со степенью свободы по указанной оси (имеет обозначение P). Ось относительного перемещения в инерциальной системе координат можно связать с одной из осей первого тела, выбрав ее из списка, либо с одной из осей второго тела (также выбрав ее из списка);
- **Resolve** – цилиндрическое сочленение, дающее вращательное движение вокруг указанной оси (имеет обозначение R);
- **Spherical** – шаровое сочленение. В этом блоке вращение возможно в любом направлении, поэтому оси не задаются. Подключение к этому блоку блока возбуждения невозможно;
- **Planar** – планарное (плоское) сочленение. При таком сочленении одна из плоскостей ведущего и ведомого тел сохраняется неизменной, при этом точки ведомого тела могут занимать любое положение в плоскости ведущего тела;
- **Universal** – универсальное сочленение, дающее возможность вращения по двум выбранным некомпланарным осям. Является последовательным соединением двух сочленений R1 и R2 типа **Resolve**;
- **Cylindrical** – цилиндрическое сочленение, обеспечивающее вращательное и поступательное движения по выбранной оси;
- **Gimbal** – сочленение типа карданового подвеса, позволяющее осуществить вращение по трем некомпланарным осям;
- **Custom Joint** – наборное сочленение, создаваемое пользователем в виде последовательной цепочки примитивов P1,2,3 типа **Prizmatic**, R1,2,3 типа Resolve и одного соединения S типа **Spherical**;
- **Weld** – жесткое сочленение. Обычно используется для создания моделей поводковых механизмов;
- **Telescoping** – телескопическое соединение, при котором одно тело по выбранной оси выдвигается из другого;
- **In-plane** – сочленение, дающее свободу относительного поступательного перемещения двух тел в плоскости указанных осей;
- **Bushing** – сочленение, дающее свободу перемещения по шести степеням свободы – три поступательных и три угловых перемещения. Блок фактически состоит из трех последовательно включенных примитивов P1,2,3 типа Prismatic и трех примитивов R1,2,3 типа Resolve;
- **Bearing** – опорное соединение;
- **Six-DoF** – блок, дающий шесть степеней свободы подобно блоку **Bushing**, в котором для обеспечения трех угловых степеней свободы используется сочленение Spherical;

- **Screw** – винтовое соединение с дополнительным параметром Pitch (шаг винта).

Помимо этих 15 блоков, данный раздел библиотеки содержит два подраздела:

- **Disassembled Joints** – разобранные соединители (рис. 12.20 слева);
- **Massless Connectors** – безынерционные соединители (рис. 12.20 справа).

Назначение блоков достаточно очевидно из их названий и обозначений.

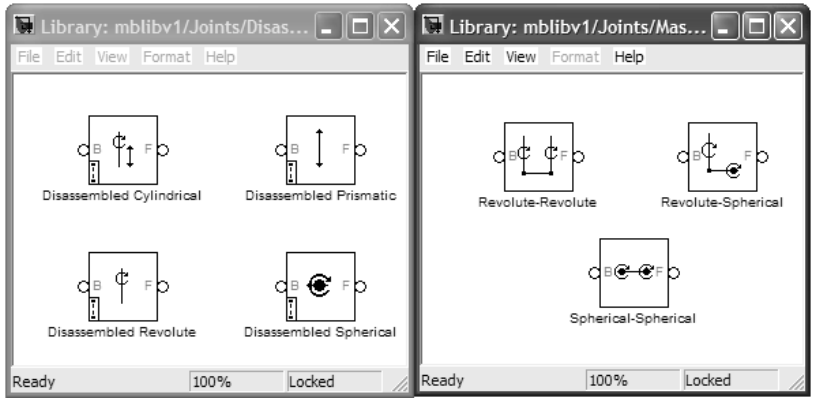


*Рис. 12.20. Блоки подразделов библиотеки **Joints***

12.4.2. Блоки раздела Sensors & Actuators

Раздел библиотеки **Sensors & Actuators** (окно на рис. 12.3 сверху и справа) содержит два типа блоков:

- **Sensors** – блоки для измерения относительного перемещения тел;
- **Actuators** – блоки – возбудители дополнительных движений.

Эти блоки позволяют связывать блоки пакета **SimMechanics** с обычными S-блоками Simulink и других пакетов расширения Simulink. Для подключения этих блоков к блокам библиотеки SimMechanics (разделов **Joints**, **Driver** и **Constrain**) у последних должны быть заданы дополнительные порты, что описывалось для блоков раздела **Joints**.

Рассмотрим блоки этого раздела. Блоки **Body Actuator (Возбудитель движения тела)** и **Body Sensor (Измеритель движения)** служат для подключения к блокам **Body**. Они задают и измеряют абсолютное движение жестко связанной с телом системы координат, к которой они подключены.

Окно параметров блока **Body Actuator** представлено на рис. 12.21. В этом окне указываются используемая координатная система, прилагаемый вращающий момент и прилагаемое усилие.

Окно параметров блока **Body Sensor**, показанное на рис. 12.22, имеет следующие параметры: координатную систему для измеряемого тела и координатную систему отсчета, позицию тела, скорость, угловую скорость, матрицу вращения,

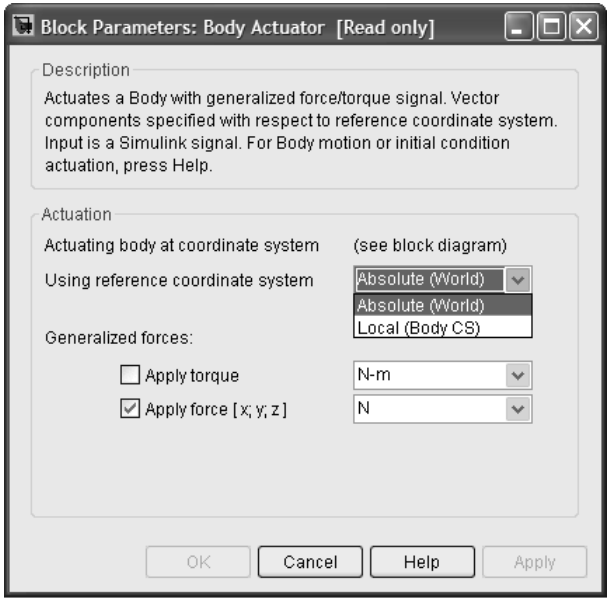


*Рис. 12.21. Окно параметров блока **Body Actuator***

ускорение и угловое ускорение. Форма задания этих параметров и единицы измерения, выбираемые из списков, видны на рис. 12.22.

На рис. 12.23 и 12.24 представлены окна параметров блоков **Joint Actuator** и **Joint Sensor** (возбудитель и сенсор сочленений). Большинство параметров этих блоков уже было описано, и в них несложно разобраться. Из новых параметров в окне сенсора сочленений (рис. 12.24) надо отметить задание кватерниона, производной кватерниона по времени и второй производной кватерниона по времени. Все они задаются 4-элементными векторами. Задаются также момент реакции относительно оси примитива и сила реакции по оси примитива.

Подобным образом представлены и окна параметров блоков **Driver Actuator** и **Constraint & Driver Sensor** (возбудитель и сенсор шкивов). Кроме того, в данный раздел библиотеки входят еще два блока:

- **Joint Initial Condition** – установка начальных условий сочленения;
- **Joint Stiction Actuator** – имитатор захвата сочленения, позволяющий учитывать силы и моменты сил вязкого и сухого трения и имитировать явление жесткого сцепления частей сочленения при наличии сухого трения.

Окно установки параметров блока **Joint Initial Condition** представлено на рис. 12.25. Установки в этом окне активны только при подключении блока к соединению. Устанавливаемые параметры вполне очевидны: флажки в позиции Enable, имена примитивов, начальные позиции с единицами измерений и начальные скорости с единицами измерений.

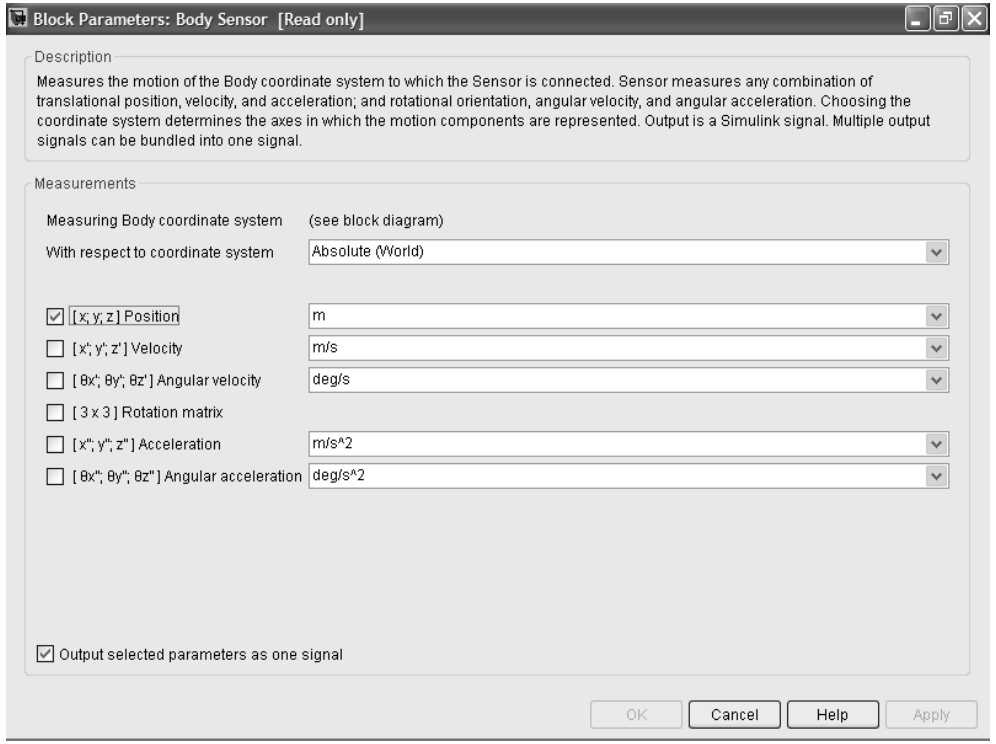


*Рис. 12.22. Окно параметров блока **Body Sensor***

Окно установки параметров блока **Joint Stiction Actuator** показано на рис. 12.26. В окне параметров этого блока задаются имя подключаемого примитива и единицы измерений внешней силы, кинетического трения и пороговой скорости.

12.4.3. Блоки раздела Constraints & Drives

Блоки раздела **Constraints & Drives** служат для имитации ограничений на степени свободы механизмов и механических систем. Окно с этими блоками показано на рис. 12.2 в левом нижнем углу. Блоки класса **Constraints** задают не зависящие от времени связи ограничения, а блоки **Drives** – ограничения, зависящие от времени движения по степеням свободы.

Конкретное назначение блоков вполне очевидно из их названий. Так, имеются следующие блоки стационарных связей:

- **Point Curve Constraints** – связь, при которой ведомое тело Follow движется по кривой, заданной узловыми точками с применением сплайновой интерполяции в его системе координат, задающей расстояния до центра системы

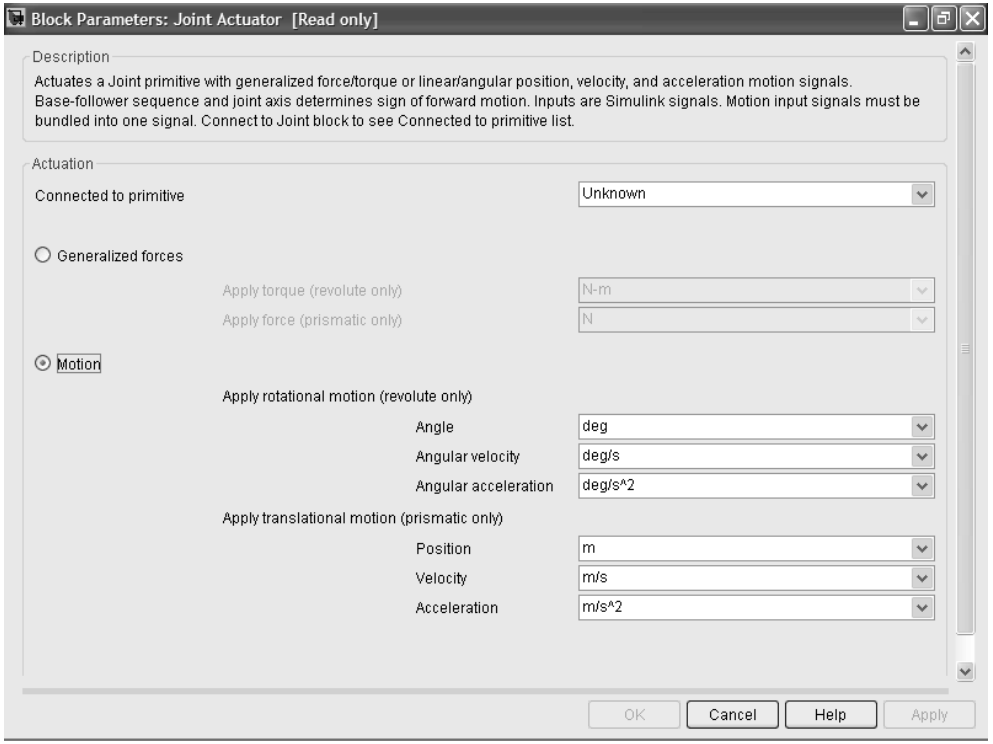


*Рис. 12.23. Окно параметров блока **Joint Actuator***

координат, связанной с ведущим телом Body (координаты точек по осям X, Y и Z задаются векторами);

- **Parallel Constraints** – связь, при которой движение тел происходит в системах координат с параллельными заданными осями (оси указываются единицей в векторе осей);
- **Gear Constraints** – зубчатая или жесткая ременная передача с заданными в блоке параметров радиусами делительных окружностей.

Кроме того, имеются четыре блока нестационарных связей. В этом случае движение задается при временной зависимости следующих величин:

- **Distance Driver** – расстояния между центрами координатных систем тел;
- **Angle Driver** – угла между заданными осями координат тел;
- **Linear Driver** – проекции расстояния между центрами координат тел на указанную ось инерциальной системы координат;
- **Velocity Driver** – линейной комбинации проекций векторов линейных и угловых скоростей координатных систем тел на указанные оси.

Поскольку установки параметров этих блоков указаны, вид окон установки параметров не приводится.

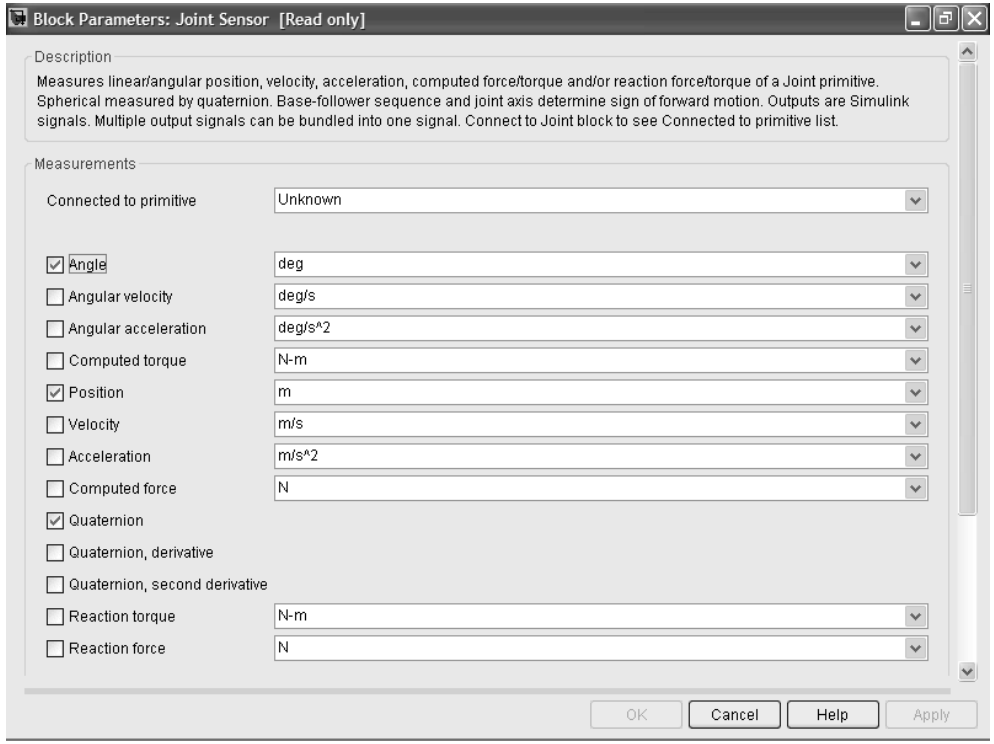


*Рис. 12.24. Окно параметров блока **Joint Sensor***

12.4.4. Блоки раздела Force Elements

В разделе библиотеки **Force Elements** (см. окно этого раздела на рис. 12.2) имеются всего два блока:

- **Body Spring & Damper** – возбудитель/амортизатор тела;
- **Joint Spring & Damper** – возбудитель/амортизатор сочленения.

Окно параметров блока **Body Spring & Damper** (рис. 12.27) содержит три основных параметра:

- **Spring constant (k)** – постоянную возбуждения;
- **Damper constant (b)** – постоянную демпфирования;
- **Spring natural length (r0)** – натуральную длину возбуждения.

Кроме того, задаются единицы измерения позиции, скорости и силы.

Окно параметров блока **Joint Spring & Damper**, показанное на рис. 12.28, содержит установки отмеченных выше для блока **Body Spring & Damper** параметров. Но они задаются несколько иначе – для каждого объекта (примитива), представленного многострочным массивом. Из этих примитивов образуется составленное сочленение.

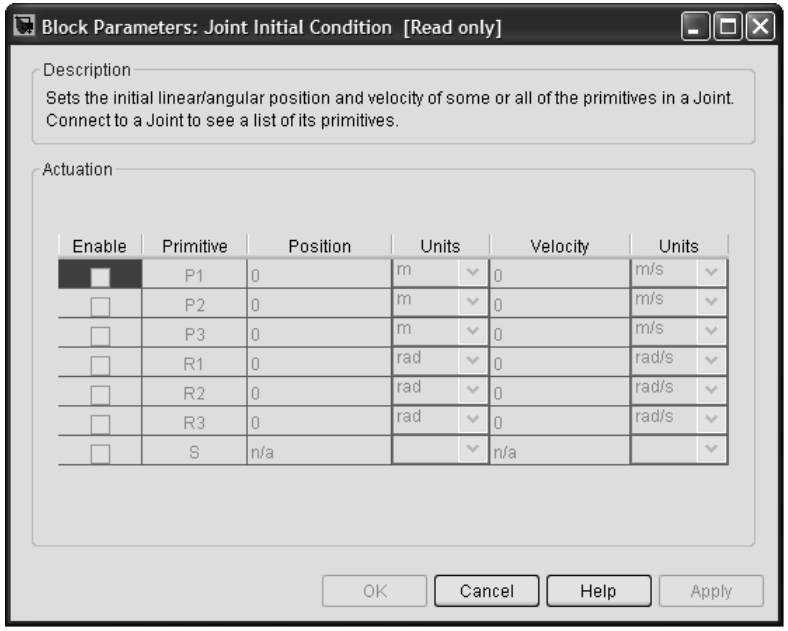


Рис. 12.25. Окно параметров блока ***Joint Initial Condition***

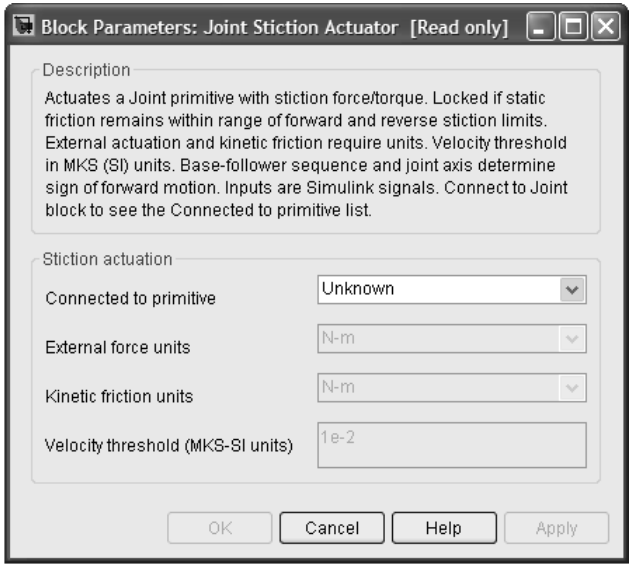


Рис. 12.26. Окно параметров блока ***Joint Stiction Actuator***

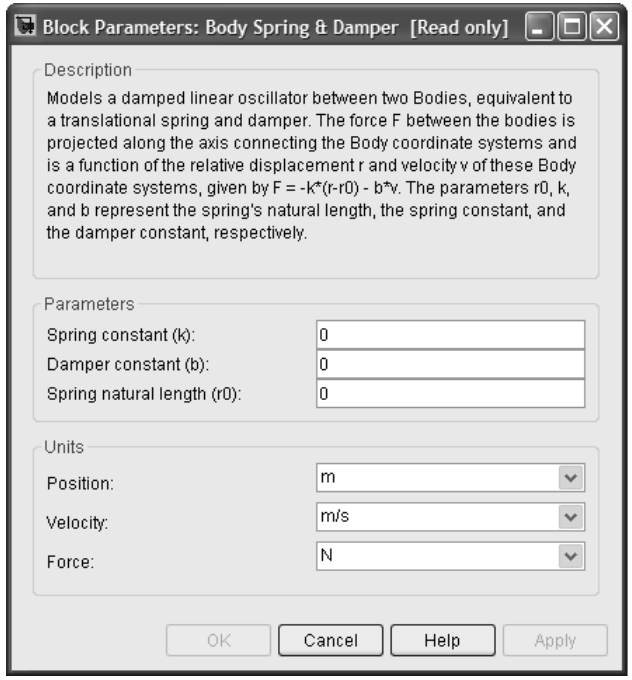


Рис. 12.27. Окно параметров блока ***Body Spring & Damper***

12.4.5. Блоки раздела Utilities

Раздел **Utilities** содержит несколько вспомогательных, но важных блока, представленных в окне, показанном на рис. 12.2 в правом нижнем углу. Всего имеются четыре блока-утилиты:

- **Connection Port** – порт, необходимый для создания субблоков (аналог портов In и Out в обычных Simulink-моделях);
- **Continuous Angle** – блок продолжения (расширения) угла;
- **Convert from Rotation Matrix to Virtual Reality Toolbox** – преобразование матрицы вращения в формат, нужный для работы со средствами виртуальной реальности;
- **Mechanical Branching Bar** – расширитель связи.

Окно блока **Connection Port** показано на рис. 12.29. Блок имеет два параметра: число портов и их расположение (слева или справа). Блок **Convert from Rotation Matrix to Virtual Reality Toolbox** имеет чисто информационное окно, установка параметров в нем не предусмотрена. Окно блока имеет единственный параметр – число разветвлений (по умолчанию 2).

Некоторого пояснения требует блок **Continuous Angle**, окно которого показано на рис. 12.30. Дело в том, что измерители угла блоков **Joint Sensor** выдают углы в

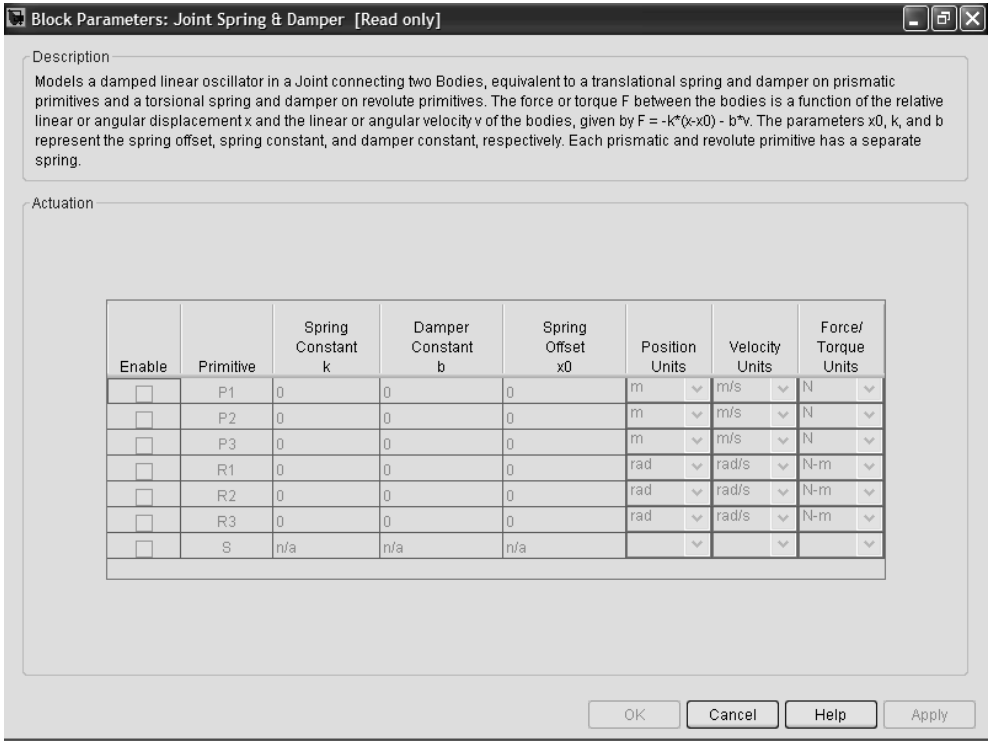


*Рис. 12.28. Окно параметров блока **Joint Spring & Damper***

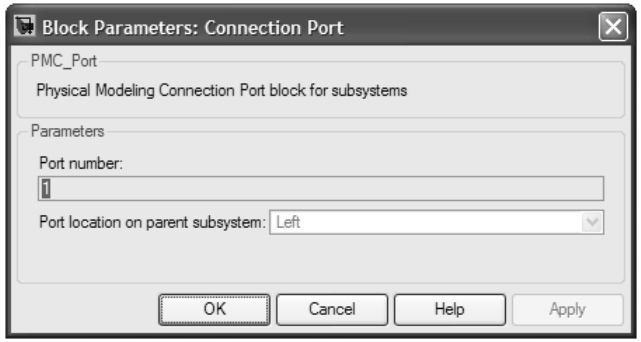


*Рис. 12.29. Окно параметров блока **Connection Port***

диапазоне от $-\pi$ до $+\pi$, и если они достигают этих значений, создается разрыв угла в 2π. Для устранения этого и служит блок **Continuous Angle**. В окне параметров данного блока устанавливается единица измерения углов (градусы или радианы),

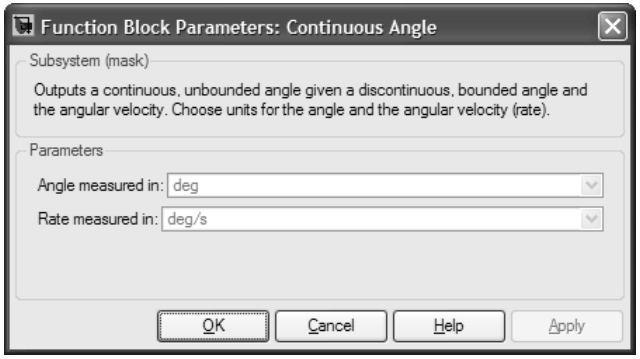


*Рис. 12.30. Окно параметров блока **Continuous Angle***

а также скорость изменения угла (угол в единицу времени). Для этой величины блок имеет порт **Rate**.

12.5. Обзор обычных демонстрационных примеров

Ниже представлено краткое обзорное описание нескольких простых демонстрационных примеров из раздела **Demos** справки по пакету SimMechanics. Целью описания является лишь представление обширных возможностей пакета в моделировании типовых механизмов. Заинтересованный читатель может внимательно рассмотреть эти примеры и разобраться в деталях работы описанных в них механизмов.

12.5.1. Моделирование отскоков упругого шарика

Пусть в пустотелой трубе сверху находится упругий шарик. Если его отпустить, он падает и, достигнув нижней точки основания, отскакивает и подпрыгивает вверх. Из-за потери энергии при отскоке он подскакивает до меньшего уровня и затем вновь падает и т. д. Диаграмма, моделирующая движение шарика, представлена на рис. 12.31.

На рис. 12.31, помимо диаграммы, представлены: субблок моделирования силы, возникающей при ударе шарика, окно расширенной визуализации с изображением прыгающего шарика и осциллограммы виртуального осциллографа. Наблюдение отскоков шарика и осциллограммы осциллографа показывают, что шарик испытывает множество отскоков с уменьшающейся высотой подъема, прежде чем он окажется лежащим на основании.

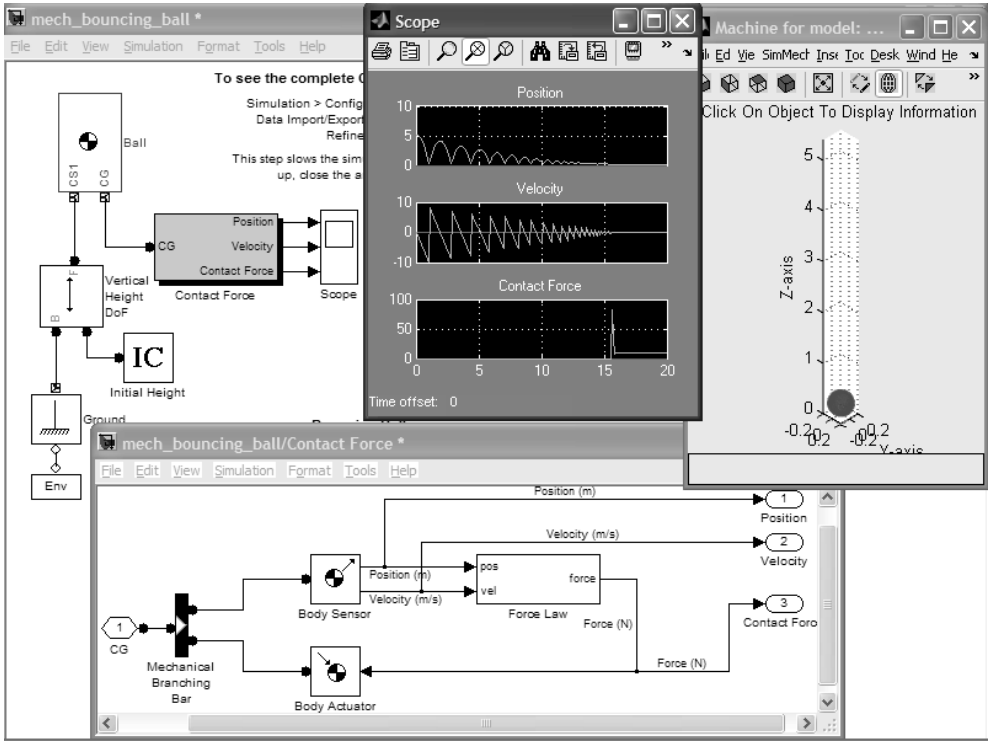


Рис. 12.31. Моделирование отскоков упругого шарика

12.5.2. Моделирование маятника с двумя стержнями

Мы уже рассматривали поведение простого маятника с одним стержнем, закрепленным с помощью цилиндрического шарнира сверху и со свободным концом снизу. Рассмотрим более сложный пример, когда снизу стержня на подвидном шарнире закреплен второй стержень. Модель такого маятника, совершающего сложные колебания, представлена на рис. 12.32.

Колебания в данном случае носят более сложный характер, особенно если собственные частоты стержней отличаются. Это и видно при пуске этого примера – на рис. 12.32 показаны осциллограммы колебаний и движения стержней в окне улучшенной визуализации.

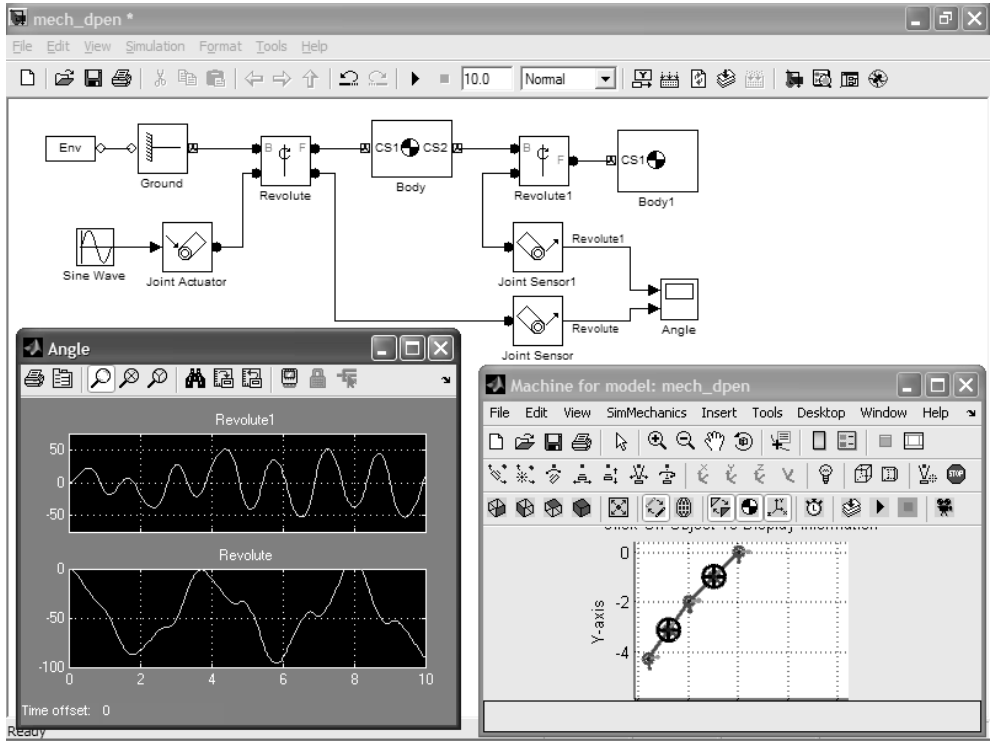


Рис. 12.32. Моделирование маятника из двух шарнирно соединенных стержней

12.5.3. Моделирование четырехзвенного маятника

Диаграмма четырехзвенного сложного маятника (механизма) представлена на рис. 12.33. Назначение этого механизма заключается в создании периодических крученых колебаний. Об их форме свидетельствуют осциллограммы, представленные на рис. 12.33.

12.5.4. Моделирование простого винтового механизма

На рис. 12.34 представлена диаграмма простого релаксационного механизма, в основе которого лежит винтовое соединение. Фактически здесь представлены два механизма, при этом верхний использует винтовое соединение. Колебания имеют пилообразную форму с разным периодом. Их временные зависимости отражают приведенные на рис. 12.34 осциллограммы.

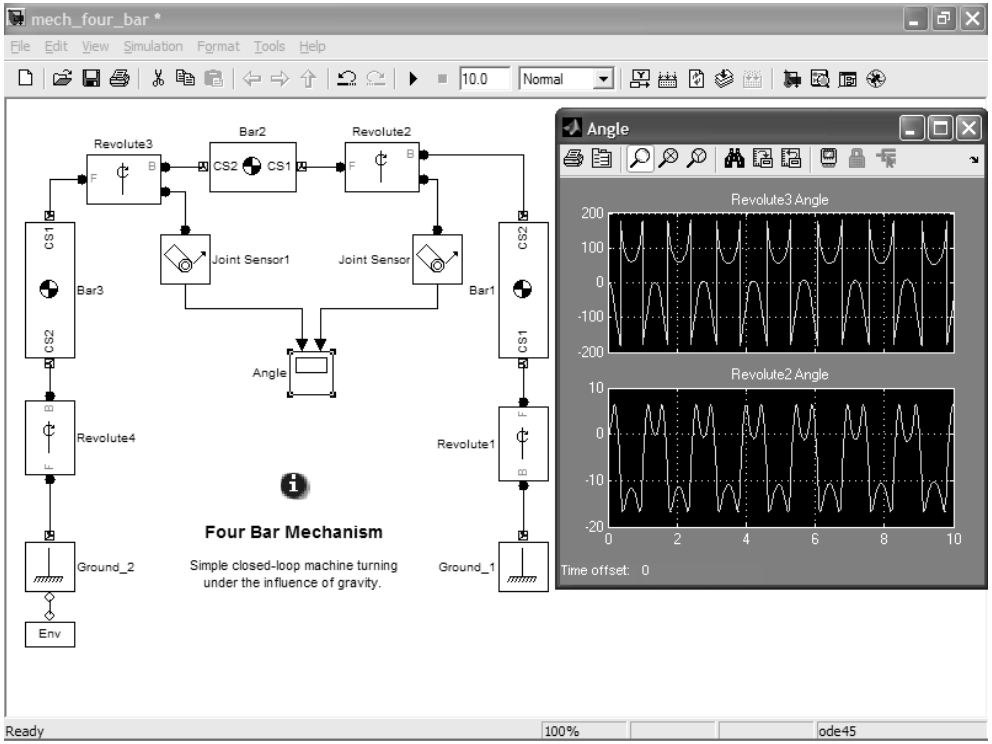


Рис. 12.33. Моделирование маятника из двух шарнирно соединенных стержней

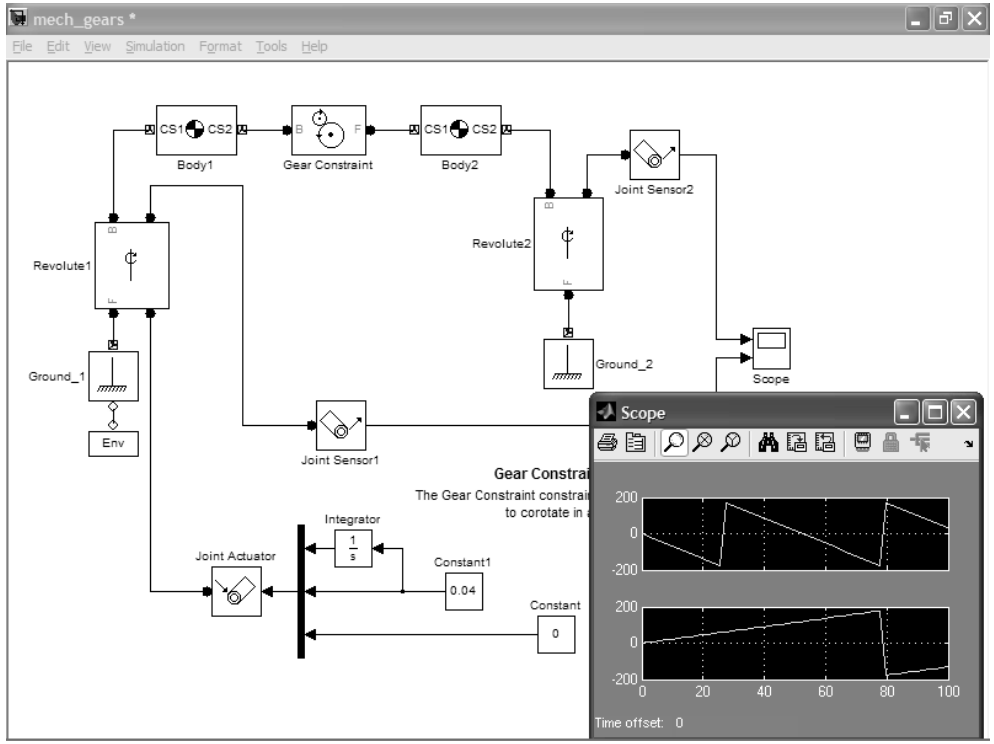


Рис. 12.34. Моделирование простого винтового механизма

12.5.5. Моделирование полета тела (баллистическая задача)

Пусть тело брошено вверх под некоторым углом и с некоторой скоростью. Диаграмма, моделирующая полет тела, представлена на рис. 12.35. Для построения траектории полета тела в данном случае используется блок графопостроителя. Траектория весьма близка к параболической, что свидетельствует о малом влиянии сопротивления воздуха на полет тела. Это справедливо, если масса тела достаточно велика, а скорость полета мала.

12.5.6. Моделирование движения тела по заданной кривой

На рис. 12.36 представлена еще одна простая диаграмма механизма, в котором имитируется движение тела по некоторой сложной линии – пространственной спирали. Такую возможность обеспечивает блок **Point-Curve Constraint**. Окно расширенной визуализации решения наглядно показывает движение тела по спирали.

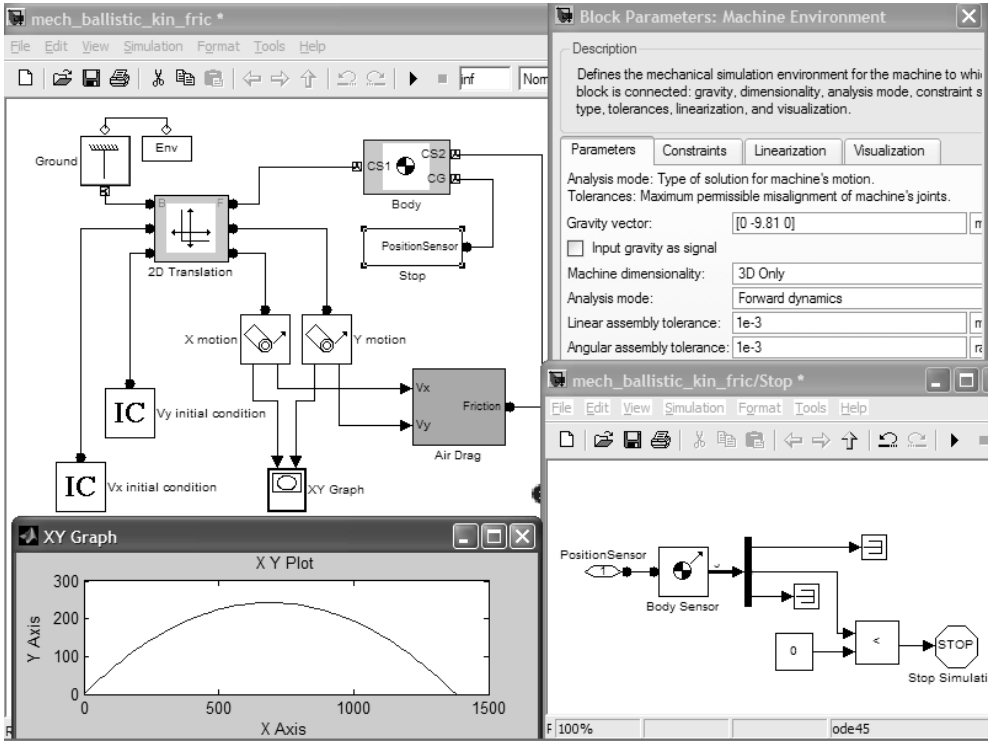


Рис. 12.35. Моделирование полета тела

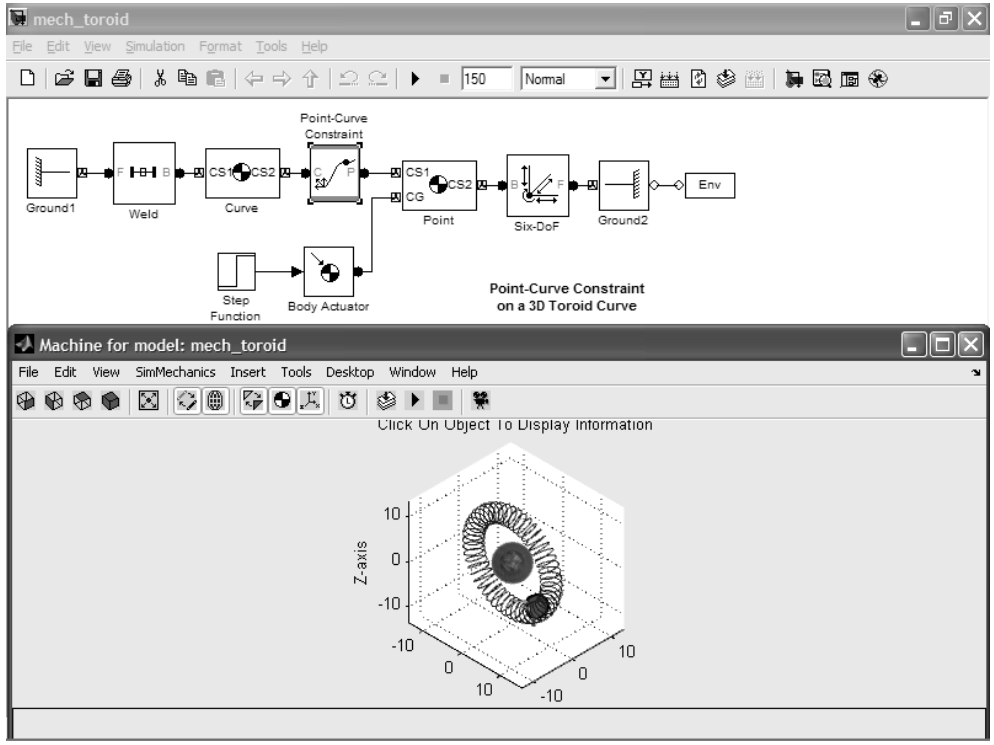


Рис. 12.36. Моделирование движения тела по заданной кривой

12.6. Моделирование механизмов с применением средств виртуальной реальности

12.6.1. Моделирование движений робота

Как уже не раз отмечалось, отличительной чертой пакета расширения **SimMechanics Blockset** является визуализация моделей в стиле, присущем механическим системам. В пакете имеется множество наглядных примеров, многие из которых прекрасно иллюстрируют это правило и полезны для изучения их пользователями, приступающими к самостоятельному моделированию механических устройств. Особый интерес вызывает привлечение средств виртуальной реальности VR (Virtual Reality), которые описываются в конце этого урока.

Хорошим примером такого подхода является простая модель роботы в демонстрационном примере Robot ARM (With VR scene). Картина, которая видна на экране дисплея при запуске данного примера, показана на рис. 12.37. Особенностью этой модели является привлечение средств виртуальной реальности, что делает изображение робота очень естественным (см. окно слева). В окне справа показан процесс моделирования в динамике с возможностью форматирования эскиза изображения в широких пределах.

Щелкнув мышью по изображению робота, можно вывести его диаграмму, которая также представлена в наглядном для механиков виде – рис. 12.38. Пиктограмма каждого блока дана наглядным рисунком той или иной части механизма робота, которые содержат множество сочленений. В диаграмме робота использован ряд субблоков, диаграммы двух из них представлены в правой части окна диаграммы. В диаграмме представлены также осциллограммы работы робота, создаваемые виртуальным осциллографом.

Под диаграммой модели робота имеется пиктограмма (на рис. 12.37 она не видна, поскольку находится внизу) **Virtual Reality Toolbox**. Ее активизация двойным щелчком мыши открывает окна средств виртуальной реальности, показанные на рис. 12.38. Они представлены субблоком виртуальной реальности (рис. 12.38 слева) и окном проигрывателя виртуальной реальности (рис. 12.39 справа).

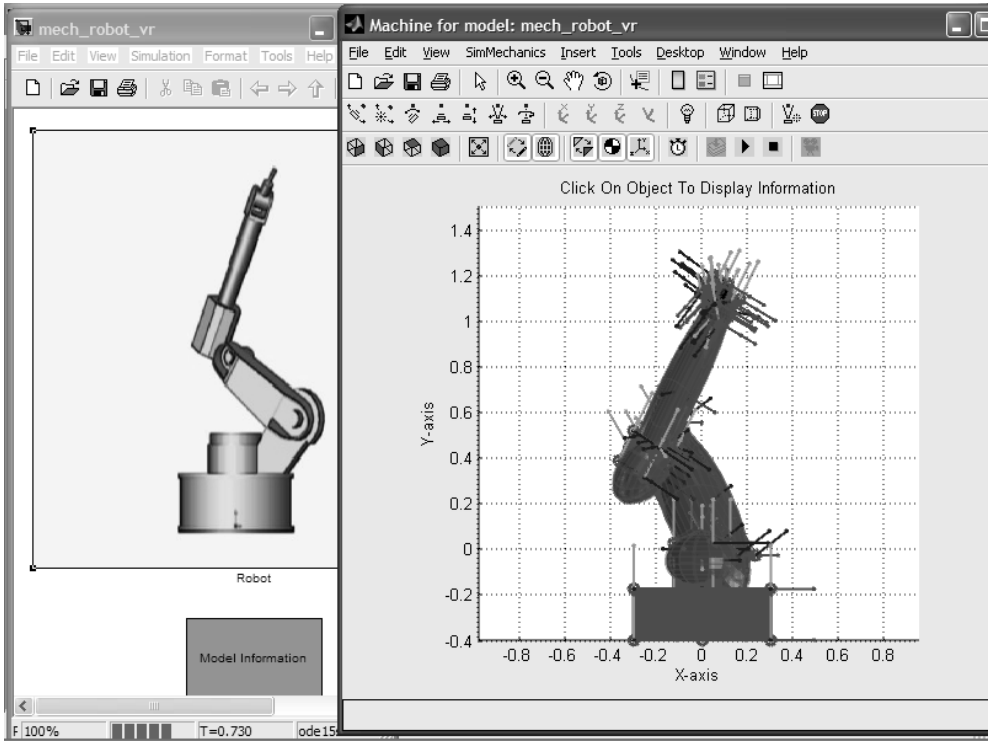


Рис. 12.37. Модель робота в механическом стиле

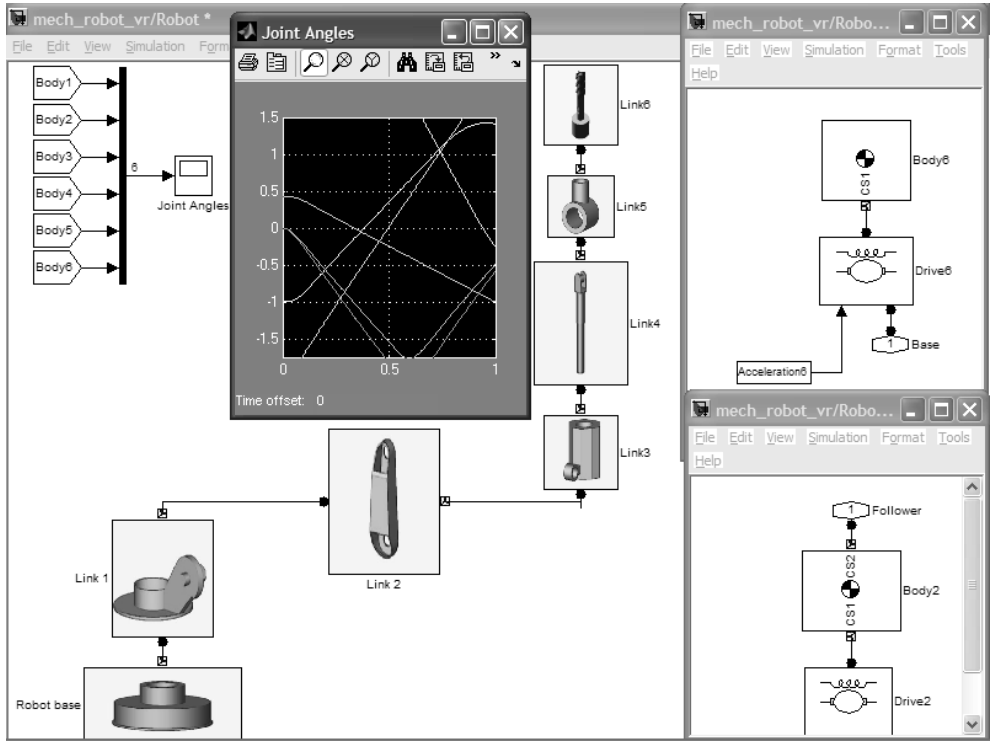


Рис. 12.38. Диаграмма модели робота (слева) и окно визуализации результатов моделирования

В окне проигрывателя виртуальной реальности видно реалистическое и действующее изображение робота.

12.6.2. Моделирование винтового планетарного механизма с виртуальной реальностью

Пример диаграммы достаточно сложного винтового планетарного механизма представлен на рис. 12.40. Разобраться с этой диаграммой под силу только достаточно опытному пользователю пакетом SimMechanics.

В правом нижнем углу диаграммы имеется пиктограмма с именем **Virtual Reality Toolbox**, открывающая доступ к средствам виртуальной реальности данного примера. Ее активизация открывает окно виртуальной реальности, показанное на рис. 12.41. Это окно при пуске диаграммы наглядно иллюстрирует работу данного довольно сложного механизма.

12.6.3. Моделирование четырехцилиндрового двигателя

Другой наглядный пример подготовки модели механического устройства показан на рис. 12.42. Здесь представлена диаграмма четырехцилиндрового двигателя внутреннего сгорания, моделирующего динамику его работы. Модель оснащена измерителем скорости вращения вала двигателя – тахометром и подготовлена в соответствии с принципами визуализации в стиле механики.

Как и в других моделях со средствами виртуальной реальности, доступ к ним дает пиктограмма **Virtual Reality Toolbox**. Окна этих средств для данного примера представлены на рис. 12.43. Нетрудно заметить, что эти средства дают хорошее представление о строении двигателя внутреннего сгорания и его работе.

Следует отметить, что описанные примеры подготовлены коллективом профессиональных разработчиков, и вряд ли их может быстро повторить пользова-

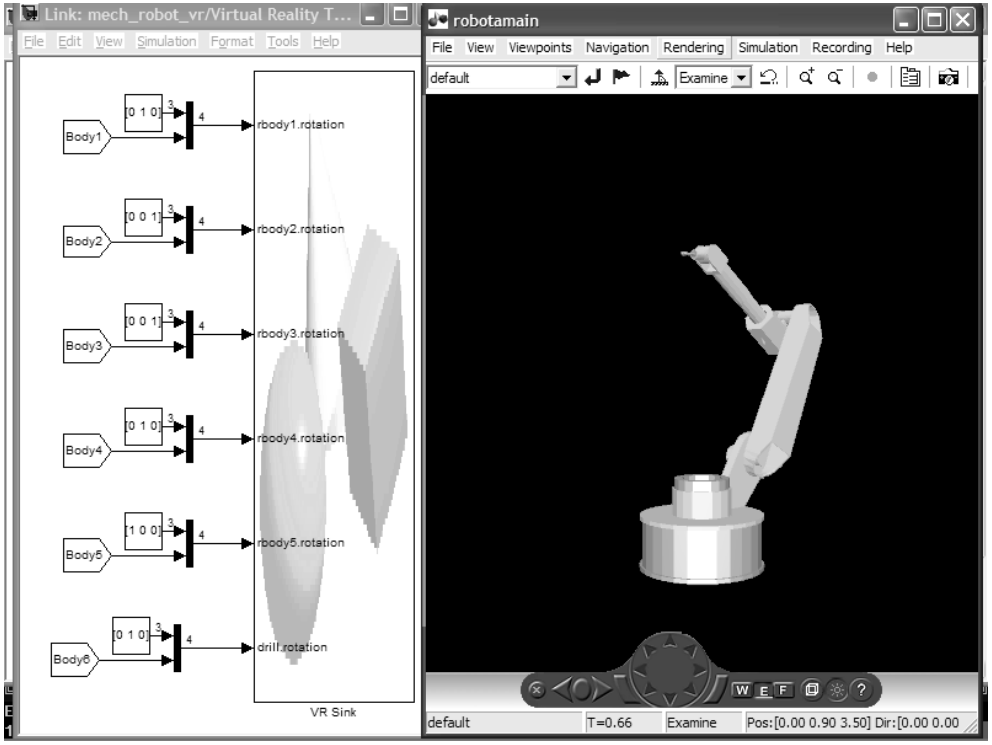


Рис. 12.39. Средства виртуальной реальности модели робота

тель пакетом **SimMechanics**. Данные примеры скорее характеризуют уровень разработки моделей в механике, к которому следует стремиться.

12.7. Пакет расширения по виртуальной реальности

12.7.1. Назначение пакета Virtual Reality Toolbox

Пакет **Virtual Reality Toolbox** служит для представления средств виртуальной реальности. Под этим громким именем скрывается совокупность средств, позволяющих представить результаты моделирования в форме, привычной для наблюдений в реальной жизни. На самом деле речь идет лишь о простейших средствах виртуальной реальности – наблюдении в анимации двумерных и трехмерных объектов.

Средства виртуальной реальности появились давно, но до появления Интернета были экзотикой. После создания Интернета они оказались востребованны-

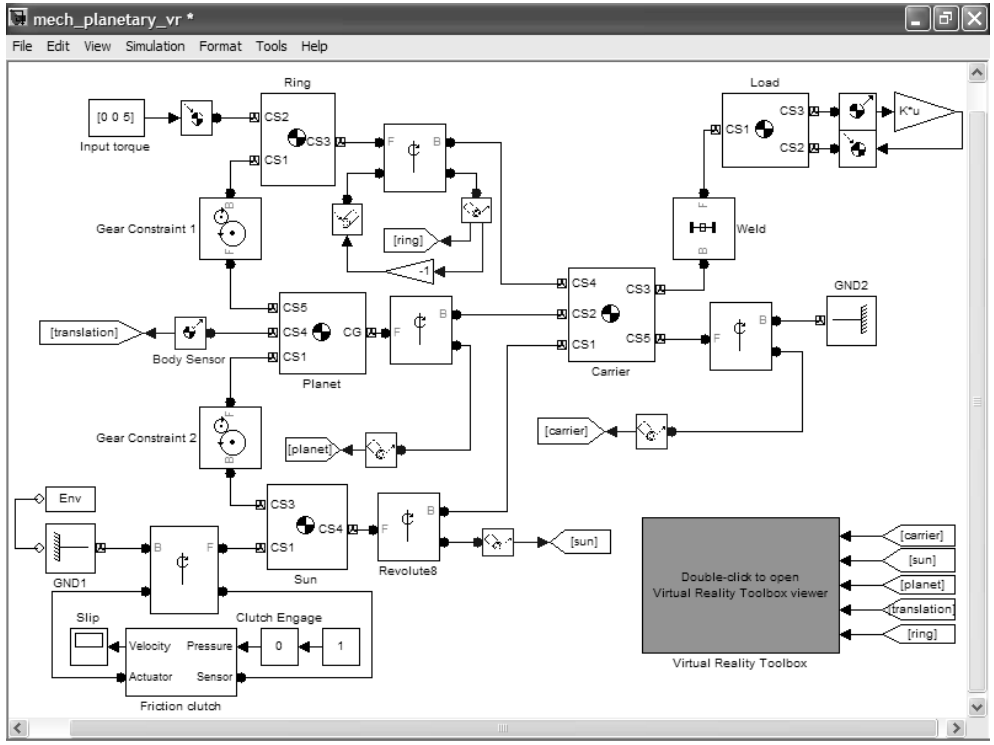


Рис. 12.40. Диаграмма сложного винтового механизма

ми. Появилась возможность с их помощью передавать данные в виде зрительных и звуковых образов на большое расстояние. Например, можно было осматривать покупаемый дом в другой стране, бродить по нему и по его окрестностям, наблюдая виды комнат и окружающие дом ландшафты.

Вскоре был создан язык моделирования виртуальной реальности Virtual Reality Modeling Language (VRML) и приняты различные его спецификации. Одна из них VRML97 лежит в основе средств пакета Virtual Reality (RT). Тут уместно сразу отметить отличия в координатных системах, принятых для трехмерной графики в MATLAB и в VRML. Они представлены на рис. 12.44.

В системе MATLAB 6.5 SP1 + Simulink 5 использована версия пакета Virtual Reality Toolbox 3.1 (сокращенно VR). В новейшей версии MATLAB 7 + Simulink 6 используется уже версия пакета Virtual Reality Toolbox 4.0 (и 4.2.1 в реализации MATLAB 7 SP3). Однако в пределах представленного здесь обзорного описания особых различий между этими версиями практически нет.

Пакет **VR** обеспечивает следующие основные возможности, в основном по части графики:

- **Virtual Worlds** – создание виртуальных миров по стандарту VRML;

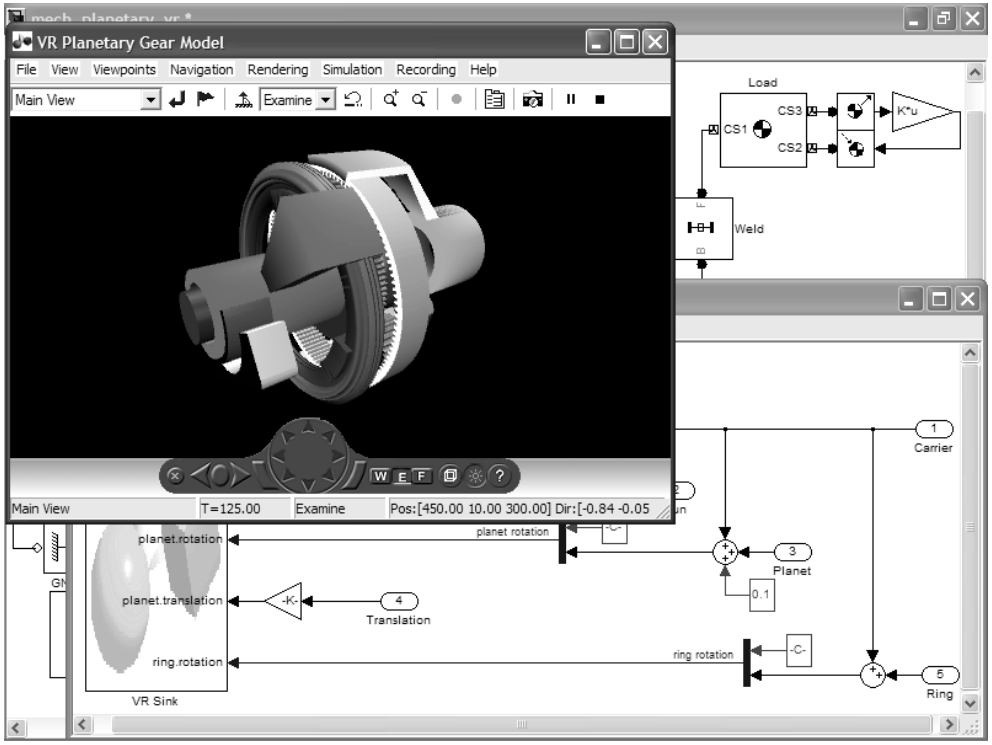


Рис. 12.41. Окно проигрывателя виртуальной реальности сложного винтового механизма

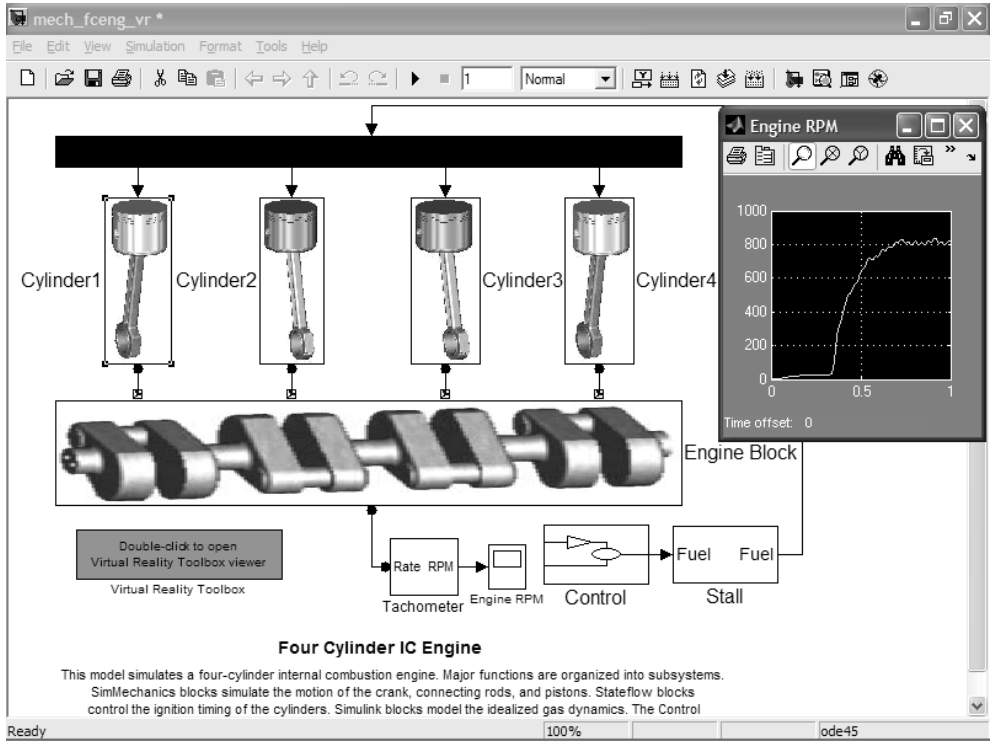


Рис. 12.42. Диаграмма четырехцилиндрового двигателя внутреннего сгорания

- **Dynamic Systems** – построение динамических систем на основе программной среды MATLAB + Simulink;
- **Animation** – предоставление расширенных средств анимации;
- **Manipulation** – осуществление различных манипуляций с объектами.

Детальное знакомство с пакетом возможно по его справке. Доступ к разделу справки по этому пакету представлен на рис. 12.45.

Пакет расширения **VR** служит для реализации средств виртуальной реальности в системе MATLAB. Для этого он имеет соответствующий набор функций. Кроме того, в состав пакета входят два важных и мощных компонента его GUI:

- **VRLM Viewer** – вьювер для просмотра моделей виртуальной реальности;
- **VRLM Editor** – редактор для подготовки моделей виртуальной реальности.

С их помощью можно удобно работать со средствами виртуальной реальности и включать их в свои программные продукты. Пакет расширения **VR** может использоваться совместно с различными другими пакетами расширения: Simulink, Stateflow, SimMechanic, Aerospace Blockspace и др.

12.7.2. Что такое виртуальная реальность в пакете VR?

Что же реально скрывается под громкими словами «виртуальная реальность» в пакете **VR**? Чтобы судить об этом объективно, надо просмотреть несколько демонстрационных примеров на вкладке **Demos** справки. Доступ к ним показывает рис. 12.46.

Всего на вкладке **Demos** дано около полутора десятков примеров применения пакета **VR**. Вначале ознакомимся с одним из них – Car in Mountains (автомобиль в горах). Этот пример демонстрирует типичный сюжет виртуальной реальности – горный ландшафт с двумя горами-пиками с туннелями и автомобильной дорогой между ними (см. первый кадр анимации данного примера на рис. 12.47). Данный пример можно пустить и из командной строки MATLAB, выполнив команду **vrcar**.

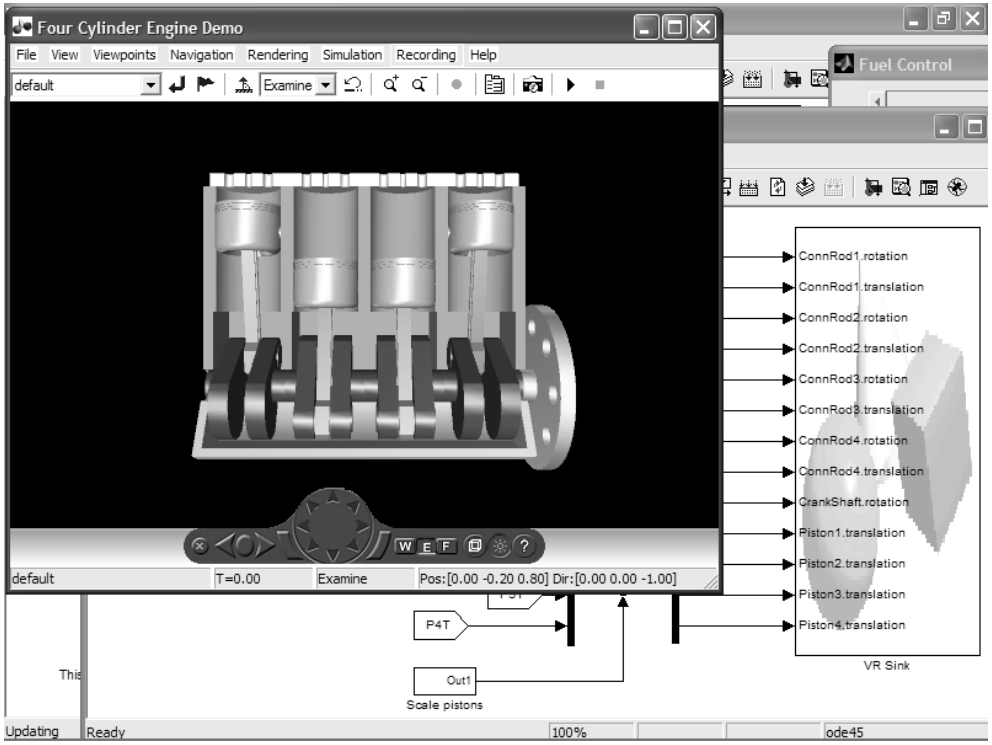


Рис. 12.43. Средства виртуальной реальности для модели четырехцилиндрового двигателя внутреннего сгорания

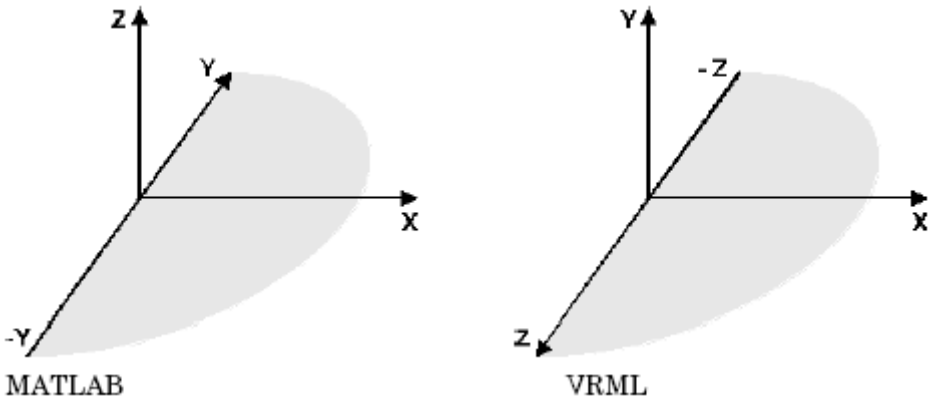


Рис. 12.44. Координатные системы MATLAB и VRML

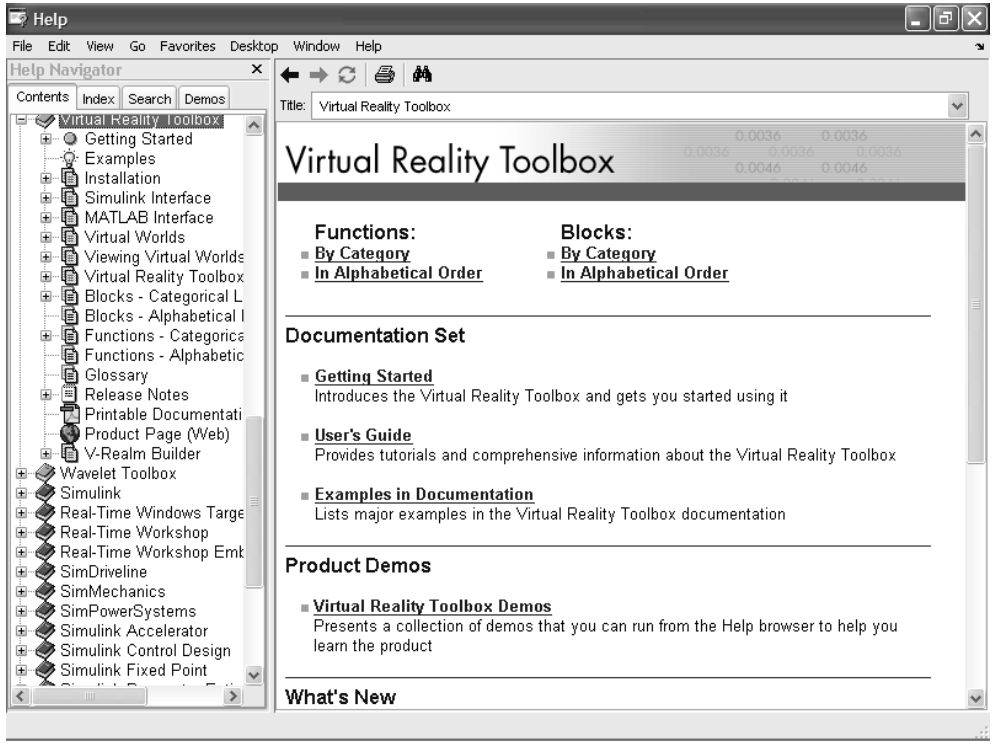


*Рис. 12.45. Доступ к справке по пакету расширения **VReality***

Основным средством работы с объектами виртуальной реальности является проигрыватель, который виден внизу рис. 12.47, на котором показано окно VR-вьювера. С его помощью можно перемещать объекты виртуальной реальности, приближать и удалять их, наблюдать каркасную модель виртуальной реальности и т. д. Учтите, что примеры выполняются в среде MATLAB, и для перехода от одной части примера к другой обычно достаточно нажать любую клавишу клавиатуры или клавишу **Enter**.

На рис. 12.48 показан промежуточный кадр данного примера. Исполняя его, вы увидите, как из туннеля в правой горе выезжает автомобиль и движется по дороге между пиками. На данном рисунке автомобиль проехал почти полпути и въезжает на мост над глубокой расщелиной. Завершается пример тем, что автомобиль скрывается, переехав через мост и въехав в туннель левого пика.

Теперь продемонстрируем возможности проигрывателя. Рисунок 12.49 показывает удаление изображения. Для этого достаточно воспользоваться графическим манипулятором проигрывателя в виде диска с треугольниками, угол которых указывает на направление перемещения. Кнопка с красной точкой возвращает

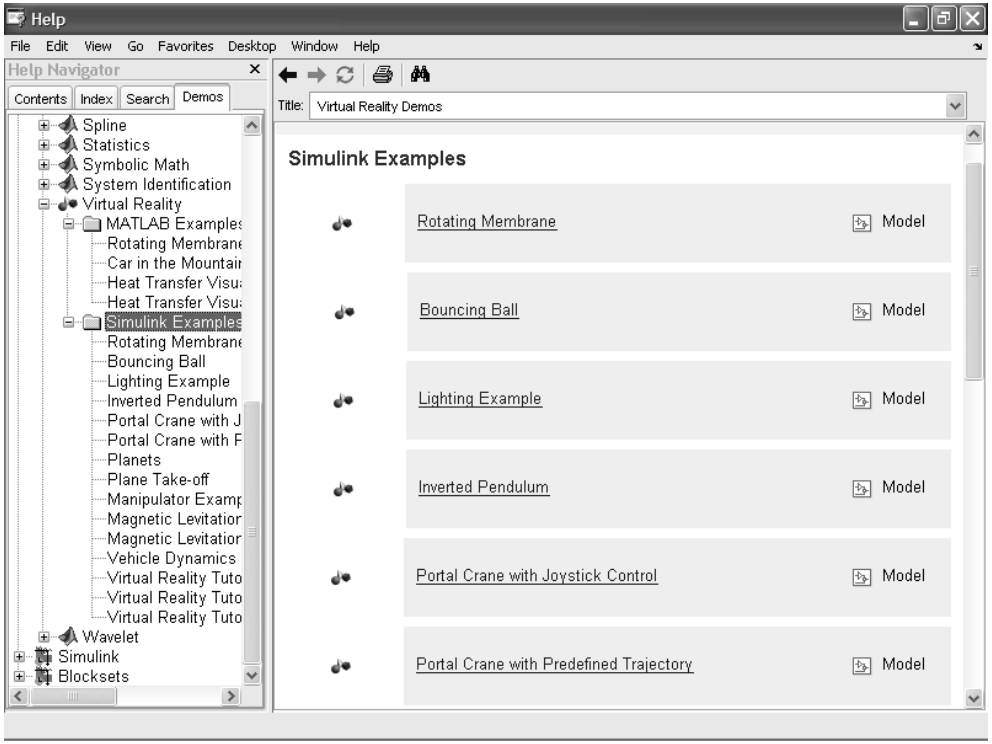


*Рис. 12.46. Доступ к демонстрационным примерам пакета расширения **VR***

изображение в исходное состояние – центрирование по центру окна просмотра VR вьювера. Эффект приближения изображения показывает рис. 12.50.

Конечно, виртуальная реальность существенно повышает степень визуализации математических расчетов и математического моделирования. Однако от средств пакета **VR** не стоит ждать чудес – к примеру, коснувшись изображения раскаленного стержня, вы не почувствуете ожога. А вожделенная для многих из нас вполне реальная баночка холодного пива в жару, извлекаемая из виртуального холодильника, – пока лишь мечта фантастов.

12.7.3. Программирование перемещения автомобиля

На первый взгляд может показаться, что приведенная выше задача чрезвычайно сложна и не под силу начинающему программисту. И это было бы так, если бы ему пришлось писать программу с чистого листа, то есть описать на языке кодов построение горного ландшафта с горами, автомобильной дорогой, мостом, самим автомобилем в разном положении и т. д.

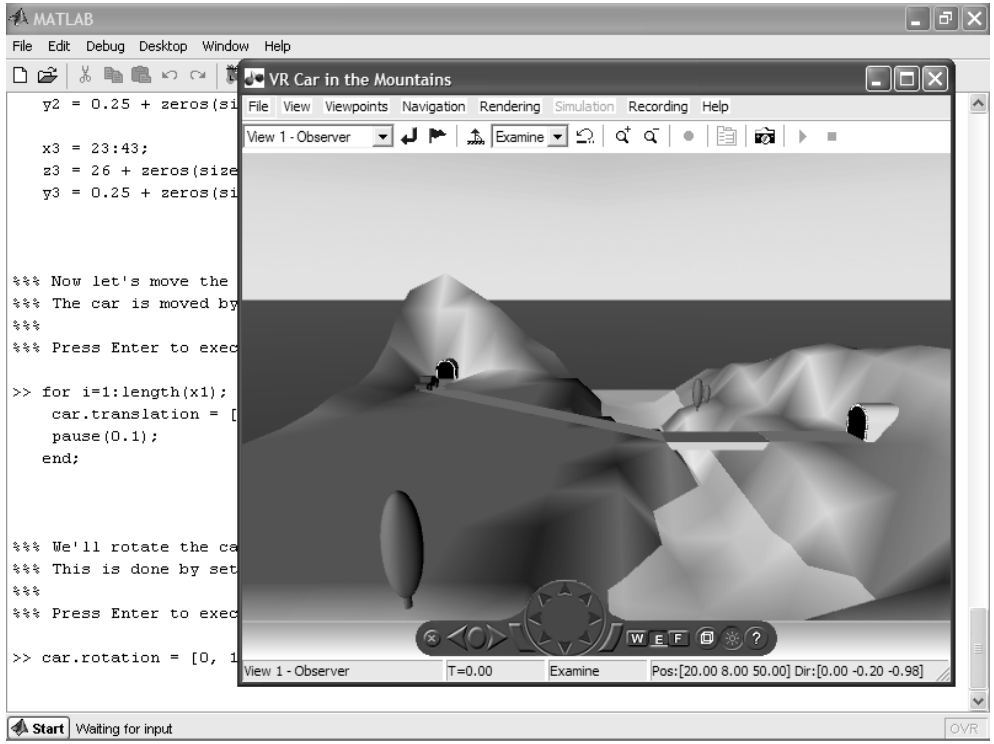


Рис. 12.47. Начальный кадр анимации из примера Car in Mountains

На самом деле все гораздо проще благодаря многочисленным заготовкам в виде команд и функций, которые уже имеются в пакете **VR**. В рамках обзорного описания пакета мы не будем рассматривать все функции этого пакета – их порядка четырех десятков для работы в командной строке MATLAB. С этими функциями и синтаксисом их записи можно познакомиться по справке пакета **VR**.

Типичным объектом пакета является виртуальный мир, который создается функцией `vrworld`, считывающей модель этого мира, заданную файлом с расширением .wrl. Например, для построения горного ландшафта со всеми его деталями достаточно исполнить в командной строке MATLAB следующие команды:

```
>> world = vrworld("vrmount.wrl")
world =
       vrworld object: 1-by-1
       (untitled) (C:\MATLAB71\toolbox\vr\vrdemos\vrmount.wrl)
>> open(world)
>> view(world);vrdrawnow
```

Появится исходный вид горного ландшафта, показанный на рис. 12.47. Теперь выведем данные об объекте:

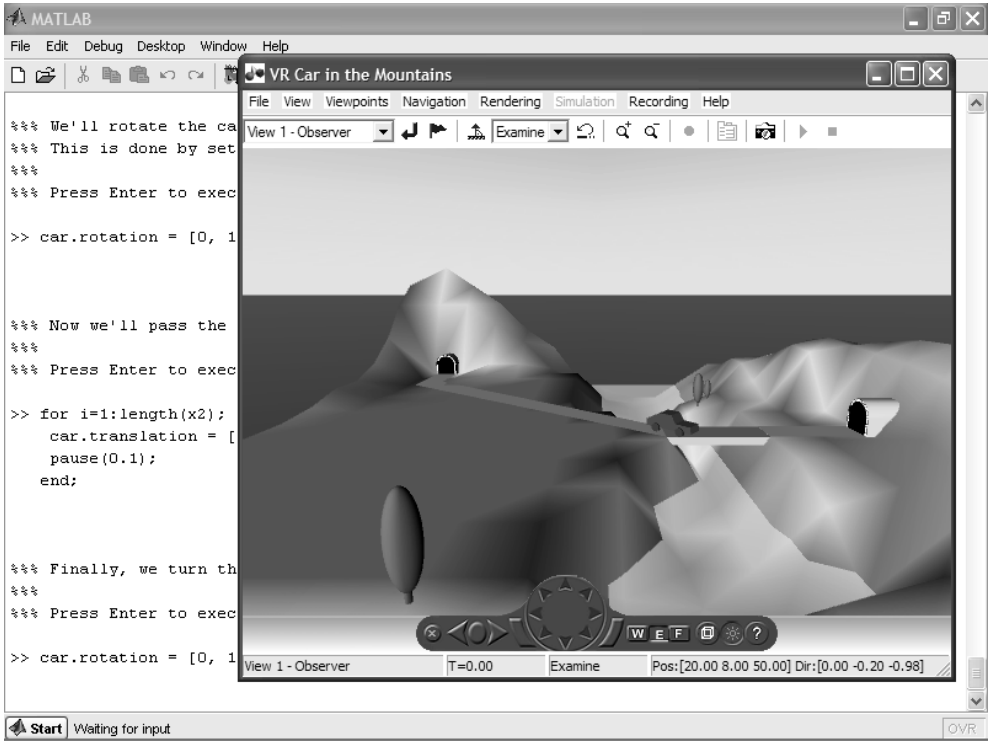


Рис. 12.48. Средний кадр анимации из примера Car in Mountains

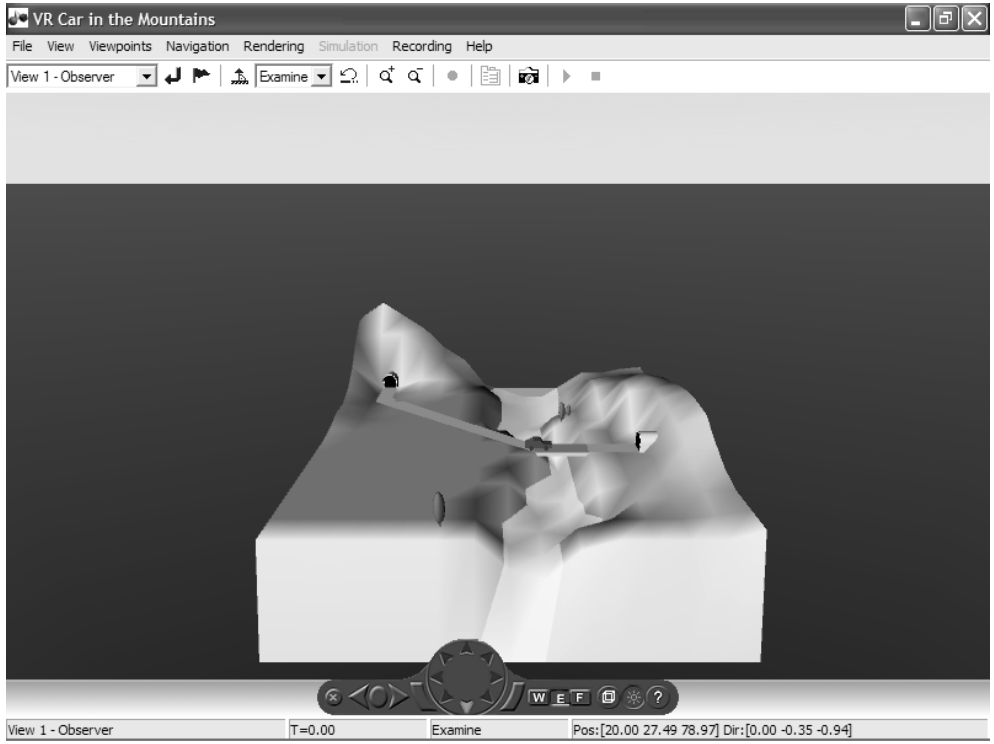


*Рис. 12.49. Действие графического манипулятора проигрывателя **VR** (удаление изображения)*

```
>> get(world)
        Clients = 1
        ClientUpdates = "on"
        Description = "VR Car in the Mountains"
        Figures = vrfigure object: 1-by-1
        FileName = "C:\MATLAB71\toolbox\vr\vrdemos\vrmount.wrl"
        Nodes = vrnode object: 13-by-1
        Open = "on"
        RemoteView = "off"
        View = "on"
```

С помощью функции `nodes` можно вывести данные об узлах объекта:

```
>> nodes(world)
        View1 (Viewpoint) [VR Car in the Mountains]
        Camera_car (Transform) [VR Car in the Mountains]
        VPfollow (Viewpoint) [VR Car in the Mountains]
        Automobile (Transform) [VR Car in the Mountains]
        Wheel (Shape) [VR Car in the Mountains]
        Tree1 (Group) [VR Car in the Mountains]
        Wood (Group) [VR Car in the Mountains]
        Canal (Shape) [VR Car in the Mountains]
        ElevApp (Appearance) [VR Car in the Mountains]
        River (Shape) [VR Car in the Mountains]
        Bridge (Shape) [VR Car in the Mountains]
        Road (Shape) [VR Car in the Mountains]
        Tunnel (Transform) [VR Car in the Mountains]
Теперь зададим новый объект — автомобиль:
>> car = vrnode(world, "Automobile")
car =
        vrnode object: 1-by-1
        Automobile (Transform) [VR Car in the Mountains]
```

Выведем детальные данные о нем с помощью функции `fields`:

```
>> fields(car)
        Field                   Access          Type            Sync
------------------------------
        translation             exposedField    SFVec3f         off
        center                  exposedField    SFVec3f         off
```

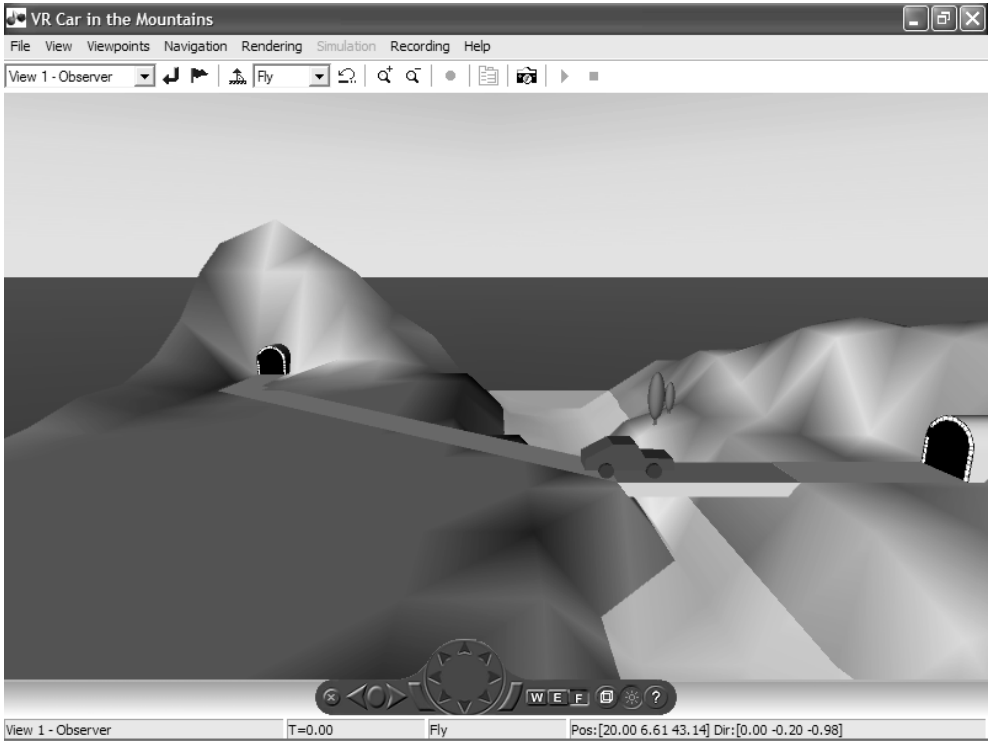


*Рис. 12.50. Действие графического манипулятора проигрывателя **VR** (приближение изображения)*

```
    bboxCenter          field          SFVec3f        off
    children            exposedField   MFNode         off
    scale               exposedField   SFVec3f        off
    bboxSize            field          SFVec3f        off
    removeChildren      eventIn        MFNode         off
    scaleOrientation    exposedField   SFRotation     off
    rotation            exposedField   SFRotation     off
    addChildren         eventIn        MFNode         off
```

Зададим характерные координаты автомобиля:

```
>>  z1 = 0:12; x1 = 3 + zeros(size(z1)); y1 = 0.25 + zeros(size(z1));
>>  z2 = 12:26; x2 = 3:1.4285:23;y2 = 0.25 + zeros(size(z2));
>> x3 = 23:43; z3 = 26 + zeros(size(x3)); y3 = 0.25 + zeros(size(z3));
```

Зададим выезд автомобиля из туннеля левого пика:

```
>> for i=1:length(x1);
    car.translation = [x1(i) y1(i) z1(i)];vrdrawnow;
    pause(0.1);
   end;
```

Появится автомобиль у выхода из туннеля левого пика – рис. 12.51.

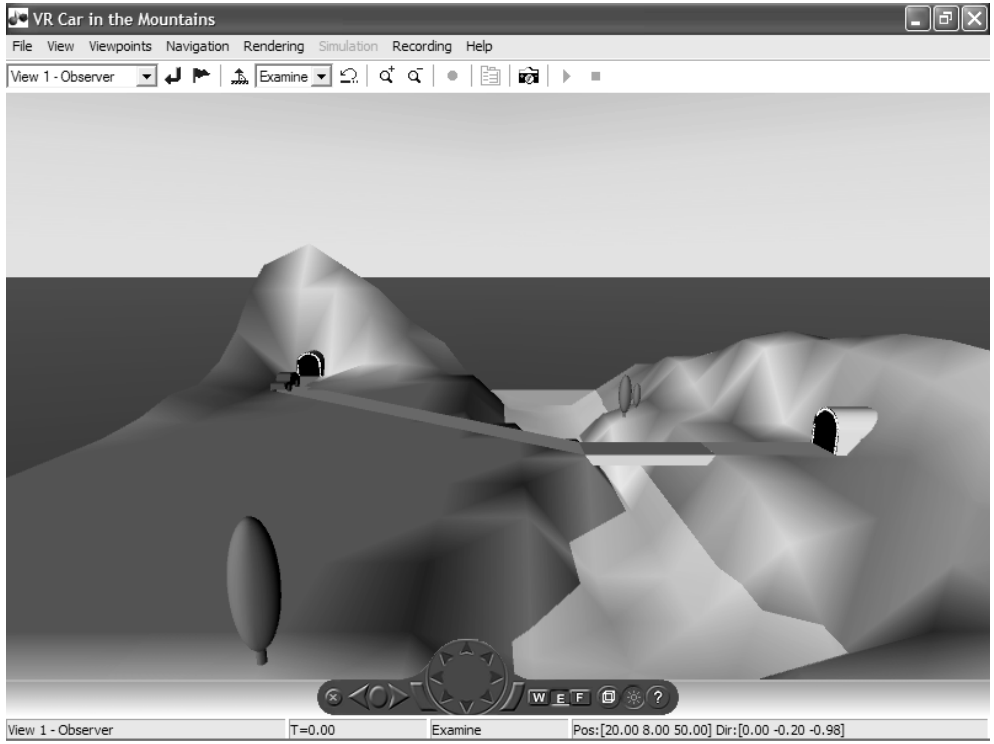


Рис. 12.51. Автомобиль выезжает из туннеля левого пика

Теперь с помощью системной переменной `car.rotation` и функции `vrdrawnow` зададим кульбит – разворот автомобиля:

```
>> car.rotation = [0, 1, 0, -0.7]; vrdrawnow;
```

Автомобиль сделает разворот – рис. 12.52.

Далее можно задать его выезд на мост:

```
>> for i=1:length(x2);
    car.translation = [x2(i) y2(i) z2(i)];vrdrawnow;
    pause(0.1);
   end;
```

Автомобиль проедет до моста и остановится у его входа – рис. 12.53.

Сделаем разворот автомобиля:

```
>> car.rotation = [0, 1, 0, 0];vrdrawnow;
```

Автомобиль почти въехал на мост – рис. 12.54. Сравните рис. 12.53 с рис. 12.54 – небольшая разница заключается в положении автомобиля. В первом случае он при движении вперед свалится с моста в глубокую расщелину, а во втором – после поворота благополучно проедет по мосту.

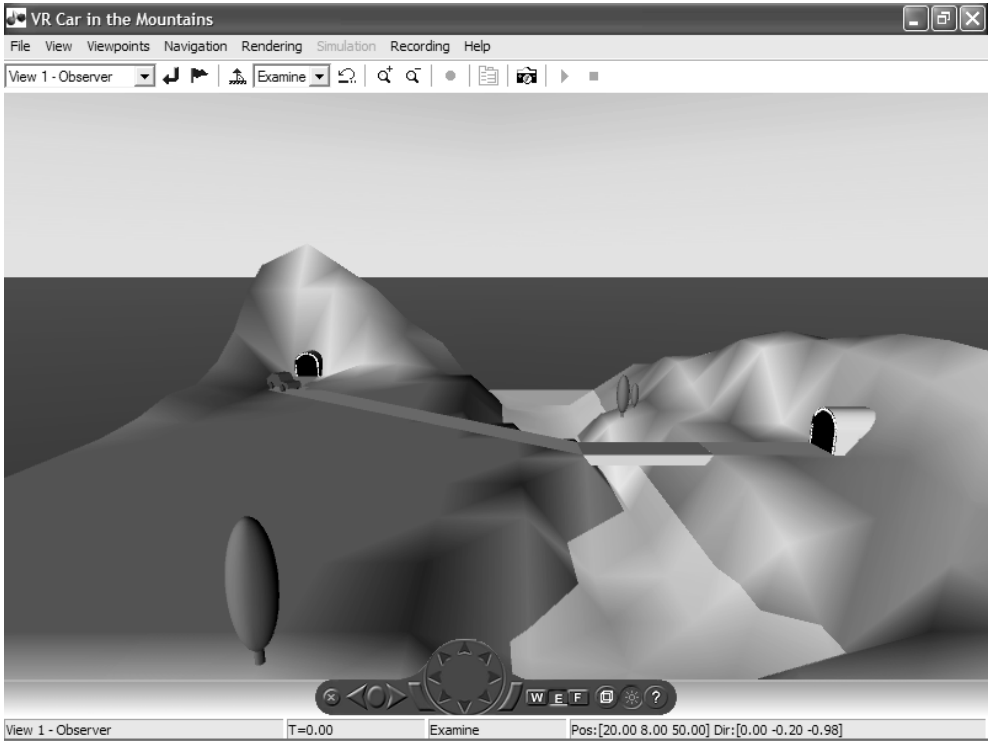


Рис. 12.52. Автомобиль делает разворот на дорогу к мосту

Теперь запрограммируем в командном режиме заключительную часть сценария – автомобиль переезжает через мост и скрывается в туннеле правого пика:

```
>> for i=1:length(x3);
    car.translation = [x3(i) y3(i) z3(i)];vrdrawnow;
    pause(0.1);
   end;
```

Автомобиль съехал с моста, переехал к правому пику и исчез в его туннеле – получим снова вид, показанный на рис. 12.47. Осталось уничтожить следы наших построений в рабочем пространстве системы MATLAB:

```
>> reload(world);vrdrawnow;
>> close(world); delete(world);
```

Итак, нетрудно заметить, что можно запрограммировать и тут же исполнить данную задачу в командном режиме. По существу, это описано в самоучителе по данному пакету (VR Tutorial), представленному выше в авторском переводе. Приведенный пример характеризует возможности пакета **VR** в командном режиме работы с системой MATLAB.

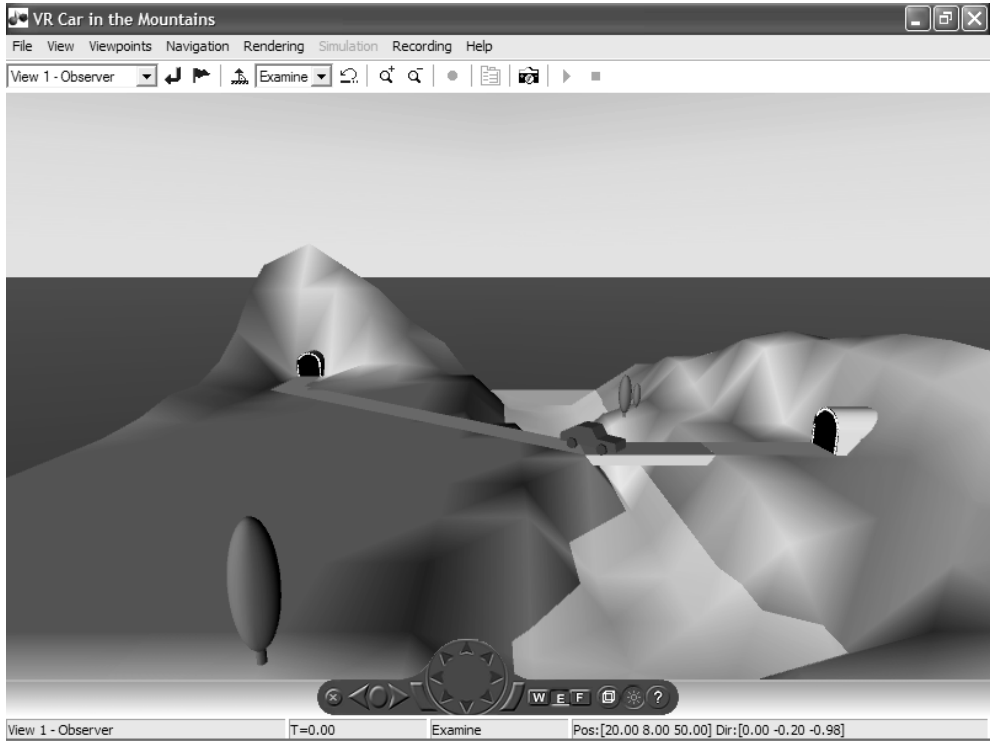


Рис. 12.53. Автомобиль выезжает к мосту

12.7.4. Блоки виртуальной реальности для Simulink

Возможности пакета **VR** существенно увеличиваются при его применении совместно с Simulink. При этом становится возможным визуально-ориентированное моделирование различных систем и устройств с наглядной визуализацией результатов средствами виртуальной реальности.

Доступ к Simulink-библиотеке средств виртуальной реальности осуществляется, как обычно, из окна вьювера библиотеки – рис. 12.55. На этом рисунке показано также открытое окно с блоками средств виртуальной реальности.

В Simulink-библиотеке средств виртуальной реальности представлено всего шесть блоков:

- **VR Sink** – VR-получатель;
- **VR Sourse** – VR-источник;
- **VR Placeholder** – администратор места;
- **VR Signal Expander** – VR-расширитель сигнала;

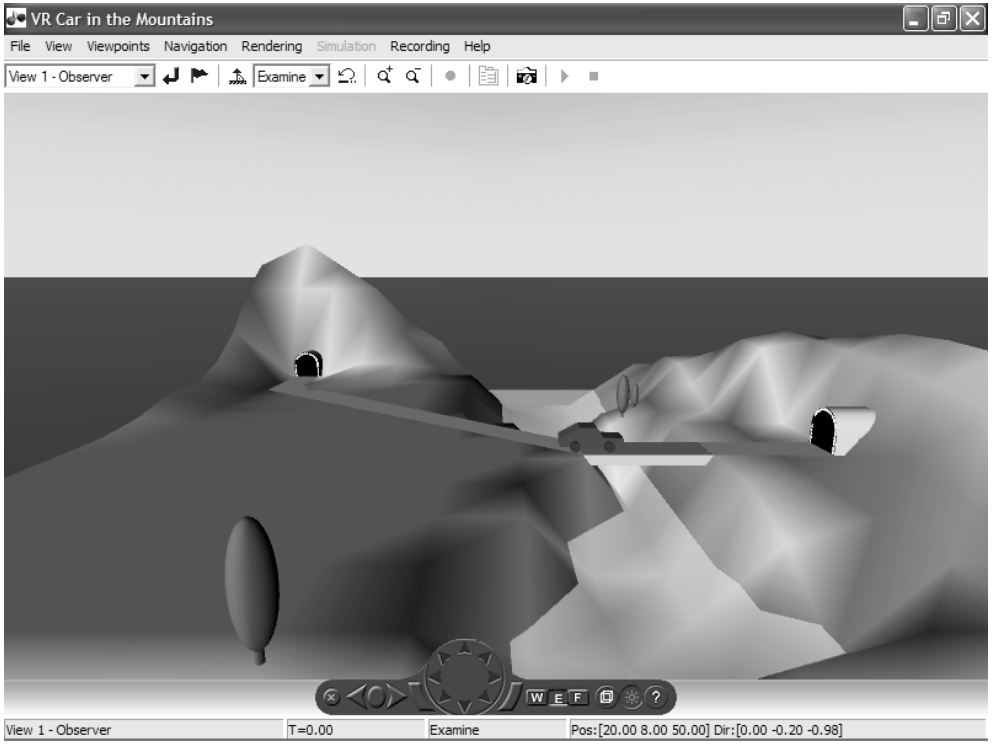


Рис. 12.54. Автомобиль въехал на мост

- **Joystick Input** – джойстик;
- **Magelan Space Mouse** – мышь Магеллана для манипуляций с трехмерными объектами.

Два последних блока можно использовать, только если ПК оснащен нужными аппаратными средствами – джойстиком и трехмерной мышью Магеллана.

12.7.5. Моделирование прыжков шара

Для начального ознакомления со средствами виртуальной реальности в справке имеется наглядный пример на визуализацию отскоков упругого шара (мяча), брошенного на твердую основу (пол). Этот пример после запуска выводит окно с Simulink-моделью, показанной на рис. 12.56.

Большая часть модели имитирует отскоки шара. Для создания виртуальности реальности здесь используются расширитель сигнала и блок получателя, который запускает VR-вьювер и позволяет наблюдать анимацию движения упругого шара. Внизу на рис. 12.56 представлены окно вьювера виртуальной реальности и осциллограммы, описывающие зависимость высоты шара от времени. Можно убедить-

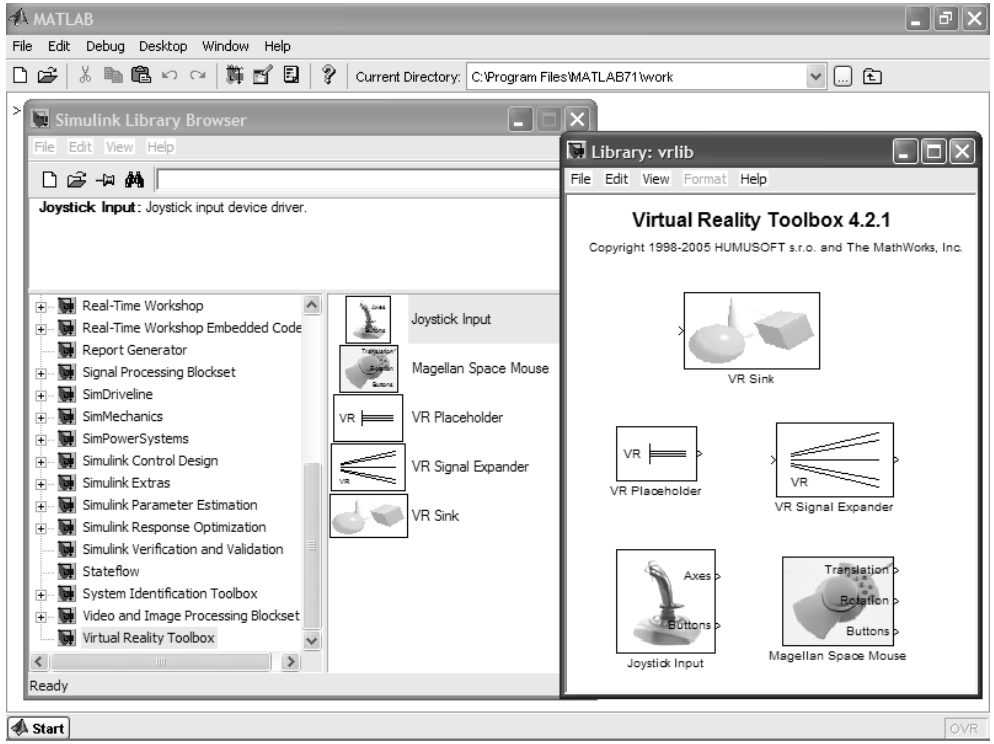


Рис. 12.55. Доступ к Simulink-библиотеке пакета виртуальной реальности

ся в том, что данная модель описывает не только положение шара, но и его деформацию в моменты удара о твердую поверхность.

12.7.6. Моделирование левитации стального шарика в магнитном поле

Процесс парения некоторого тела в пространстве без видимых внешних воздействий именуют *левитацией*. Фокусники часто используют его, ухитряясь поддерживать в свободном пространстве себя или свою ассистентку. Физики давно установили, что левитации в обычных земных условиях без воздействия на тело вообще нет, и прекрасно продемонстрировали это на примере левитации стального шарика, висящего в пространстве за счет магнитного поля.

Вполне очевидно, что с помощью электромагнита можно поднять стальной шарик с основания. Но чтобы удержать его в неподвижности в некоторой точке пространства между основанием и магнитом, надо создать специальную следящую систему, регулирующую силу магнитного поля. Как только шарик поднимается до заданной высоты, надо усилить магнитное поле, и шарик начнет набирать

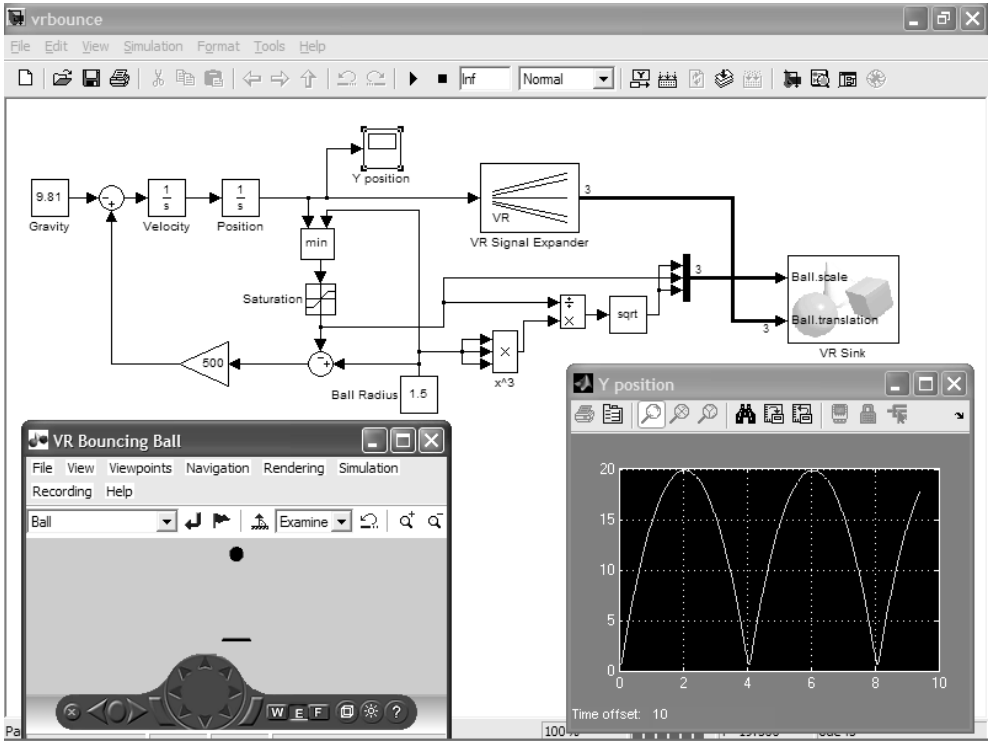


Рис. 12.56. Модель, моделирующая отскоки упругого шара от твердой основы

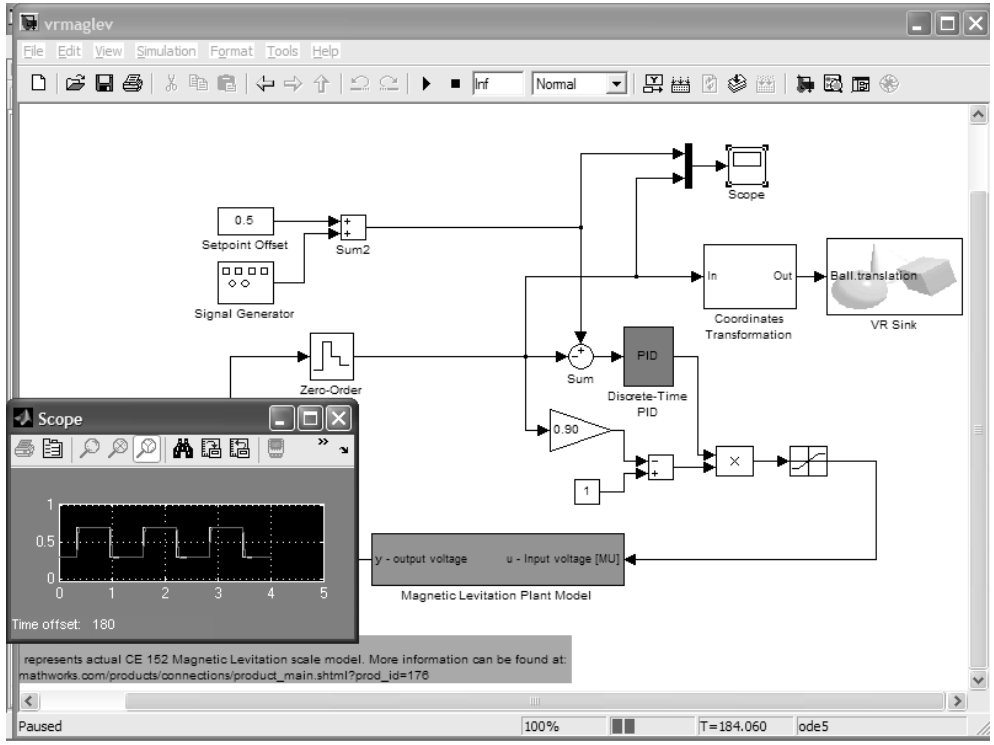


Рис. 12.57. Модель левитации стального шарика

высоту. Когда он поднимется до заданного уровня, силу магнитного поля надо уменьшить, и шарик начнет подниматься. Получаем типичную релаксационную систему, модель которой представлена на рис. 12.57.

Работа системы контролируется осциллограммами и блоком **VR Sink**, создающим окно с проигрывателем виртуальной реальности с изображением электромеханической установки левитации. Это окно показано на рис. 12.58 и дает весьма реалистическое изображение физической установки, реализующей «левитацию».

12.7.7. Пример моделирования движения автомобиля

Весьма эффектный пример моделирования движения автомобиля по шоссе можно реализовать, и применяя расширение Simulink. Этот пример представлен моделью, показанной на рис. 12.59. Блок **Coordinate Transformation** содержит субмодель движения автомобиля. Она показана на рис. 12.60.

Блок **VR Visualization** создает картину движения автомобиля, очень напоминающую реальную, – рис. 12.61. Есть возможность наблюдать движение автомобиля со стороны либо переместиться на место едущего в нем человека и наблюдать изменение окружающей обстановки при движении автомобиля.

12.7.8. Как создаются объекты виртуальной реальности

У заинтересованного читателя при внимательном рассмотрении остается пара важных вопросов. Каким образом обеспечиваются перемещение сцен и объектов, их повороты, приближения и удаления? Как вообще создаются объекты виртуальной реальности, например сцена горного ландшафта и изображение автомобиля?

Вначале ответим на первый вопрос. Любое изображение можно разбить на ячейки, напоминающие паучью сеть. Окрасив ячейки, можно получить цветное и

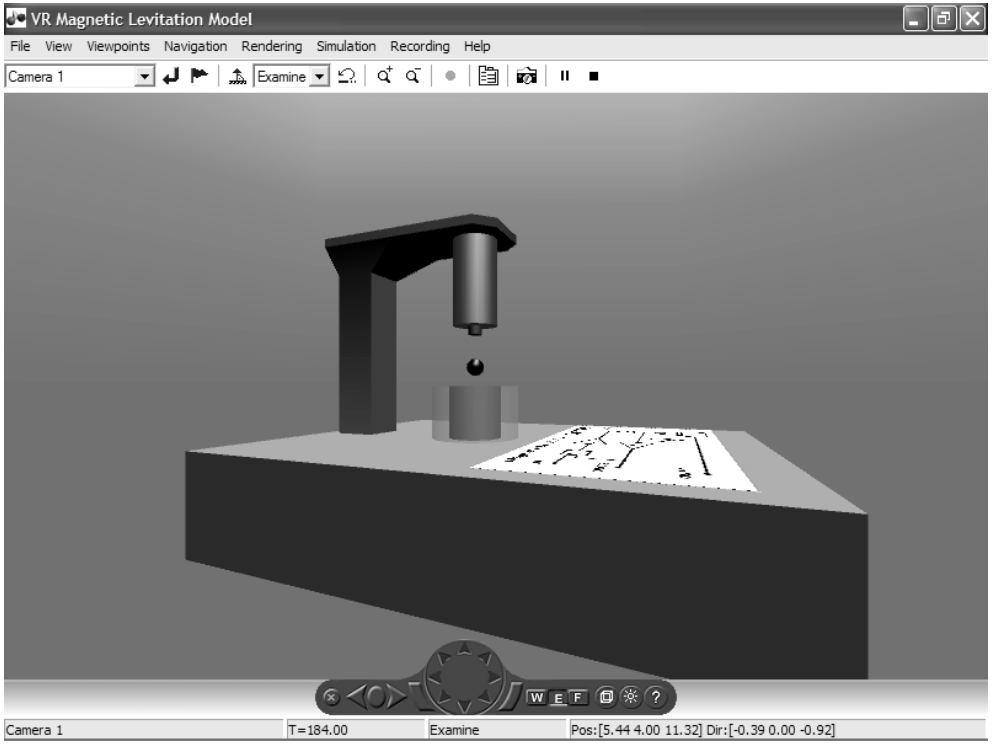


Рис. 12.58. Окно проигрывателя виртуальной ральности, демонстрирующего «левитацию» стального шарика

вполне реалистическое изображение, практически сходное (по крайней мере, на глаз). А пересчитывая матрицы узлов ячеек по специальным формулам перемещения, поворота и т. д., можно осуществлять соответствующие операции с изображением.

В пакете **VR** есть встроенные средства для осуществления этих операций. Проконтролировать их нетрудно, активизировав кнопку на панели проигрывателя виртуальной реальности с изображением каркаса куба. Появится каркас изображения, показанный на рис. 12.62. С помощью операций рендеринга производится окраска фрагментов изображения, что и создает его естественный вид.

Теперь ответим и на второй вопрос. Для создания изображений объектов виртуальной реальности служит особый редактор таких объектов, по существу являющийся разновидностью графического редактора для создания и редактирования произвольных изображений. Он вызывается из окна параметров блока **VR Sink** и показан на рис. 12.63.

В окне параметров этого блока есть кнопка **Browse** для поиска и загрузки файла, задающего изображение объекта. В нашем случае оно задано файлом vrboud.wrl

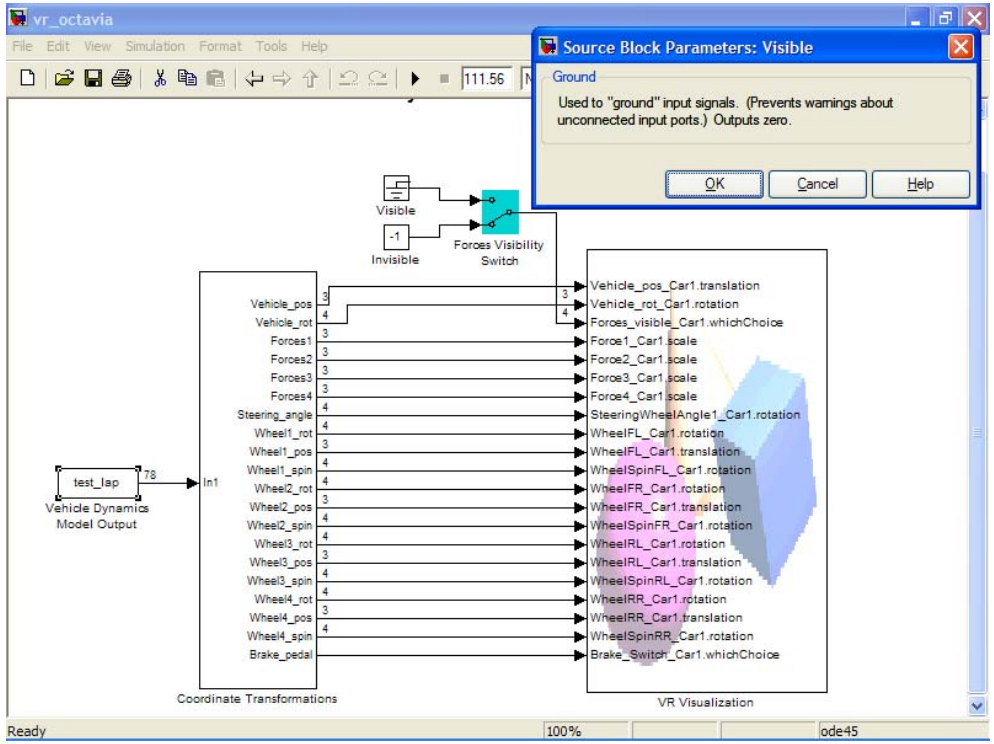


Рис. 12.59. Диаграмма модели движения автомобиля

(wrl – это расширение для файлов объектов виртуальной реальности). Для вызова объекта на редактирование в редакторе объектов **VR** достаточно активизировать кнопку **Edit** в окне параметров блока **VR Sink**. Появится окно редактора **VR** объектов, показанное на рис. 12.64.

Редактор **VR** объектов – это сложная программа с общирным интерфейсом, содержащим многие десятки кнопок. Ее подробное описание возможно только в отдельной книге и выходит за рамки этого урока. Мы ограничимся лишь общими положениями по созданию объектов виртуальной реальности с помощью этого средства.

Прежде всего надо отметить, что изображение состоит из множества различных объектов, и каждый должен быть введен с помощью тех или иных кнопок панелей инструментов. Это могут быть точки, линии, прямоугольники, окружности (окрашенные и без окраски) и т. д. и т. п. Редактор предназначен для синтеза изображения из таких объектов. При этом строится дерево объектов, показанное на рис. 12.64 слева. Созданное или отредактированное изображение записывается в виде файла с расширением .wrl.

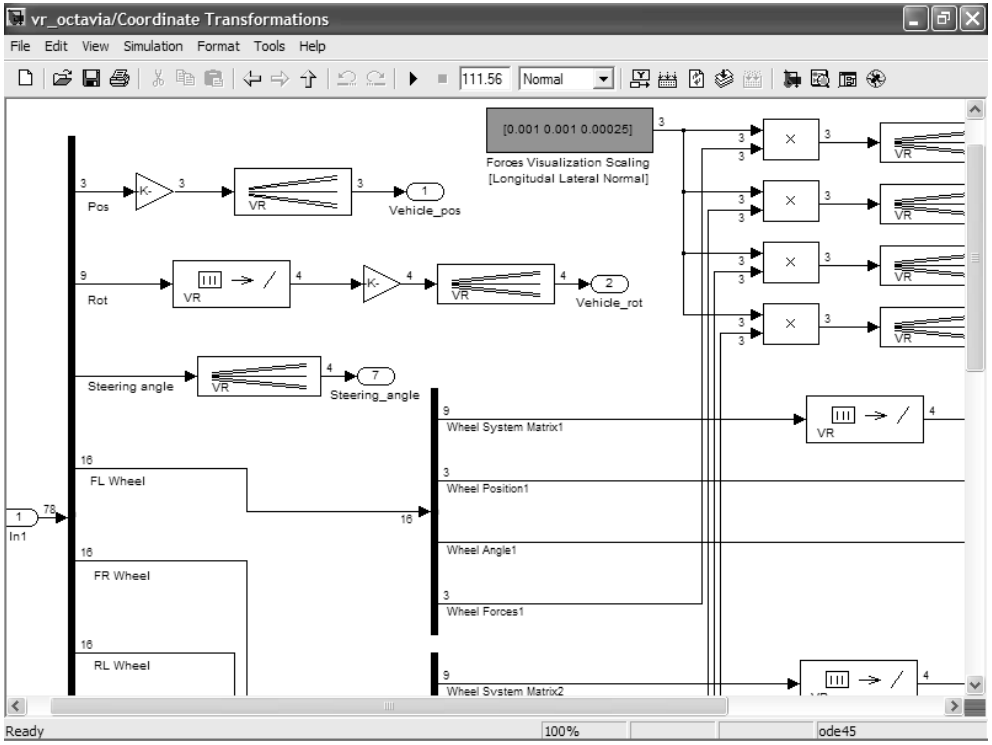


Рис. 12.60. Субмодель модели движения автомобиля

Готовое изображение объекта нетрудно редактировать, имея некоторые навыки работы с редактором. Возможна подготовка изображений с чистого листа, но это уже требует достаточно профессионального использования редактора. Тут уместно вспомнить известную поговорку – не боги горшки лепят! Заинтересованный читатель наверняка освоит работу с редактором объектов виртуальной реальности точно так же, как осваивается работа с обычными графическими редакторами.

12.8. Основы моделирования аэрокосмических аппаратов

12.8.1. Назначение пакета Aerospace Blockset и состав его библиотеки

Пакет расширения **Aerospace Blockset** предназначен для моделирования летательных, в том числе аэрокосмических, аппаратов с применением основного расширения – Simulink. Ниже обзорно рассматривается новейшая версия Aerospace

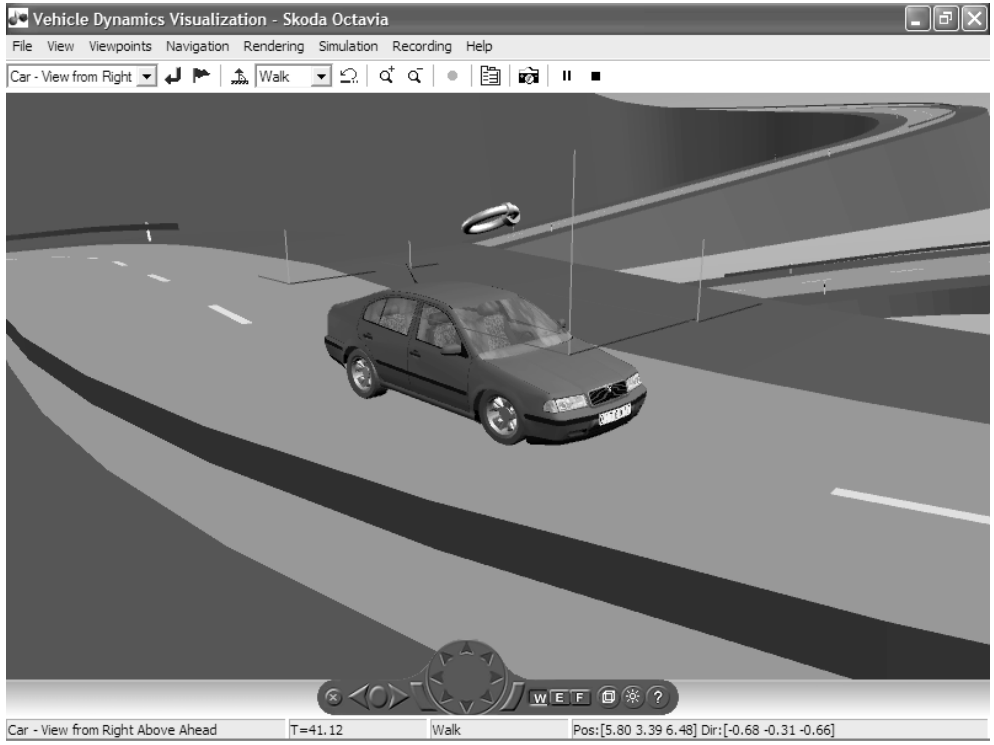


Рис. 12.61. Окно виртуальной реальности, показывающее движение автомобиля

Blockset 2.0.1, применяемая в новейшей системе MATLAB 7.1 + Simulink 6.3. Информацию об уже сильно устаревшей версии пакета Aerospace Blockset 1.0 можно найти в [ ]. *АВТОР! ГДЕ?*

Доступ к библиотеке данного пакета из окна библиотеки Simulink самый обычный – рис. 12.65. Он также возможен из командной строки MATLAB командой **aerolib**. Организация библиотеки, принципы использования блоков и задания их параметров самые обычные, те, что неоднократно рассматривались выше.

В состав библиотеки входят 10 основных разделов:

- **Equations of Motions** – блоки задания уравнений движения;
- **Propulsion** – блок моделирования влияния двигательной установки (один);
- **Actuators** – блоки моделирования поведения приводов рулей;
- **GNC** – блоки регуляторов управления движением;
- **Environment** – блоки моделирования влияния окружающей среды;
- **Aerodynamics** – блок аэродинамики (один);
- **Mass Properties** – блоки моделирования свойств масс;
- **Flight Parameters** – блоки параметров полета;
- **Utilities** – блоки утилит;

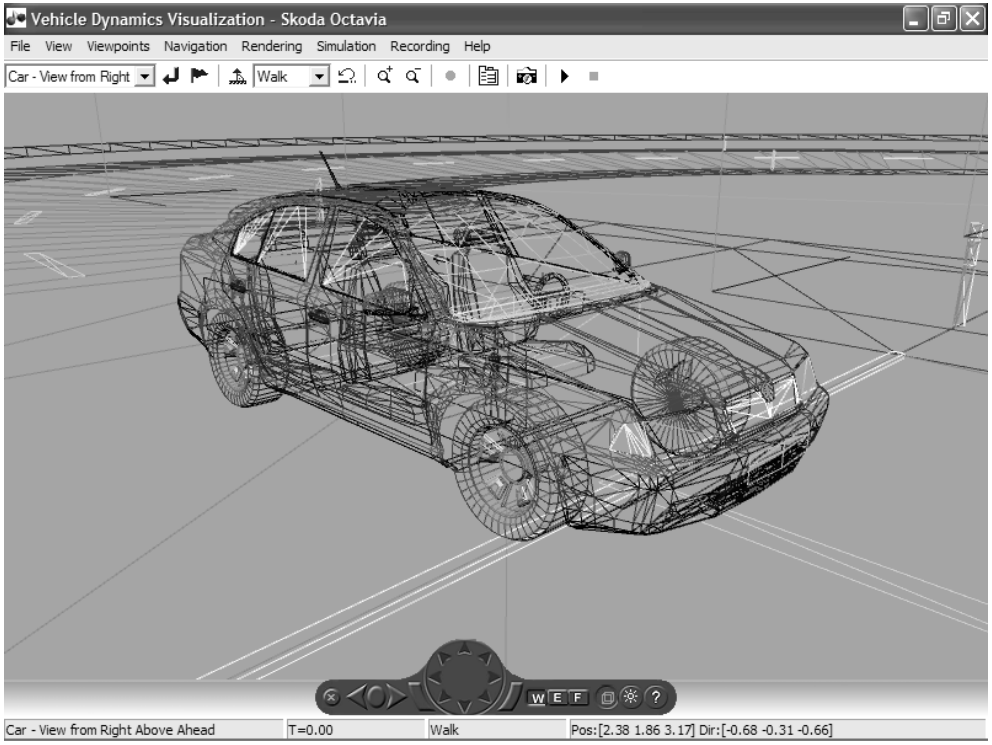


Рис. 12.62. Просмотр каркаса изображения

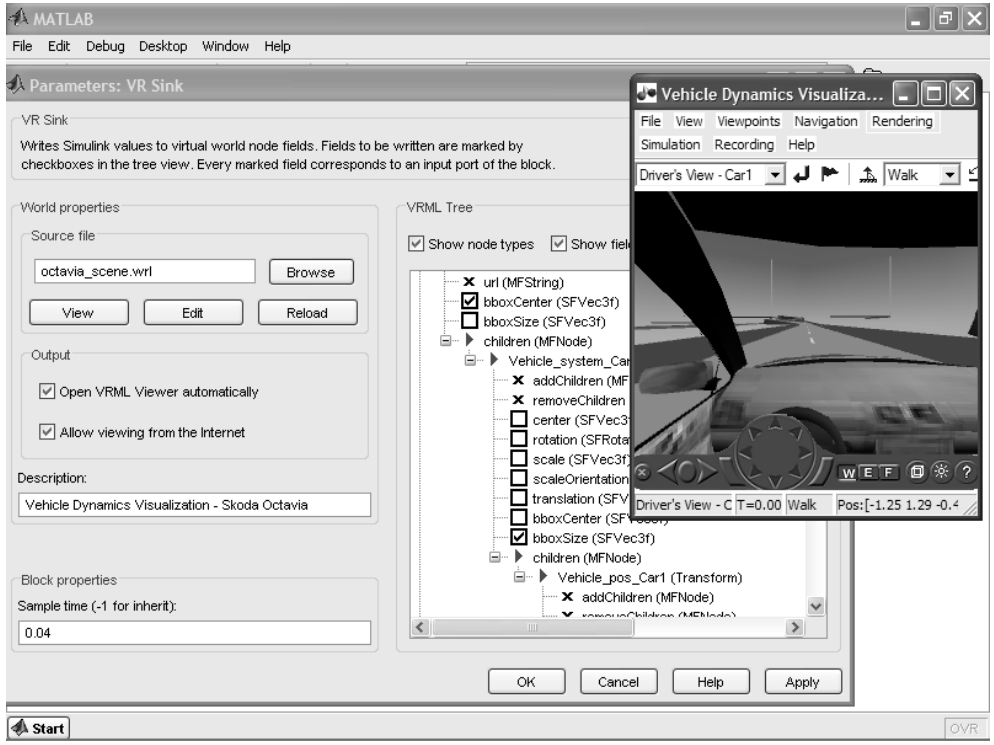


*Рис. 12.63. Окно редактора блока параметров **VR Sink***

- **Animation** – блоки анимации движения аэрокосмических аппаратов.

Кроме того, в окне основной библиотеки имеются пиктограммы **Info** – выхода в справку по пакету и **Demos** – выхода в раздел демонстрационных примеров справки. Окна большинства разделов этой библиотеки представлены на рис. 12.66.

12.8.2. Координатная система пакета Aerospace Blockset

Для понимания сути многих понятий, относящихся к пакету **Aerospace Blockset**, нужно учитывать особенности координатных систем этого пакета. За основную систему координат (она же является инерциальной) принята система координат с осями X_e, Y_e и Z_e, привязанными к поверхности Земли. При этом ось Z_e направлена вниз к центру Земли, а две другие оси лежат в плоскости горизонта. Кривизна поверхности Земли и вращение последней не учитываются, так что моделироваться может только движение тел вблизи поверхности Земли.

Вторая система координат с осями X_b, Y_b и Z_b по умолчанию имеет центр в центре масс O тела – летательного аппарата. Направление осей следующее:

- X_b – продольная ось, направленная к носу летательного аппарата;
- Y_b – перпендикулярная ей ось в плоскости крыла и направленная вправо, если смотреть с хвоста аппарата в его нос;
- Z_b – перпендикулярная плоскости крыла ось, направленная вниз.

Проекции вектора скорости летательного аппарата $\boldsymbol{V}$ на его оси X_b, Y_b и Z_b обозначаются как u_b, v_b и w_b, а проекции вектора абсолютной угловой скорости $\boldsymbol{\omega}$ аппарата – как p, q и r. Проекции вектора момента внешних сил M, действующих на летательный аппарат, обозначаются как L, M и N.

В параметрах ряда блоков фигурируют углы Эйлера:

- ψ (yaw) – угол рыскания (угол отклонения в плоскости горизонта продольной оси X_b летательного аппарата от направления оси X_e основой системы координат);
- θ (pitch) – угол тангажа (угол подъема продольной оси летательного аппарата над плоскостью горизонта);
- φ (roll) – угол крена (поворота летательного аппарата вокруг его продольной оси).

Типичное положение летательного аппарата (ракеты) в пространстве представлено на рис. 12.67. Здесь:

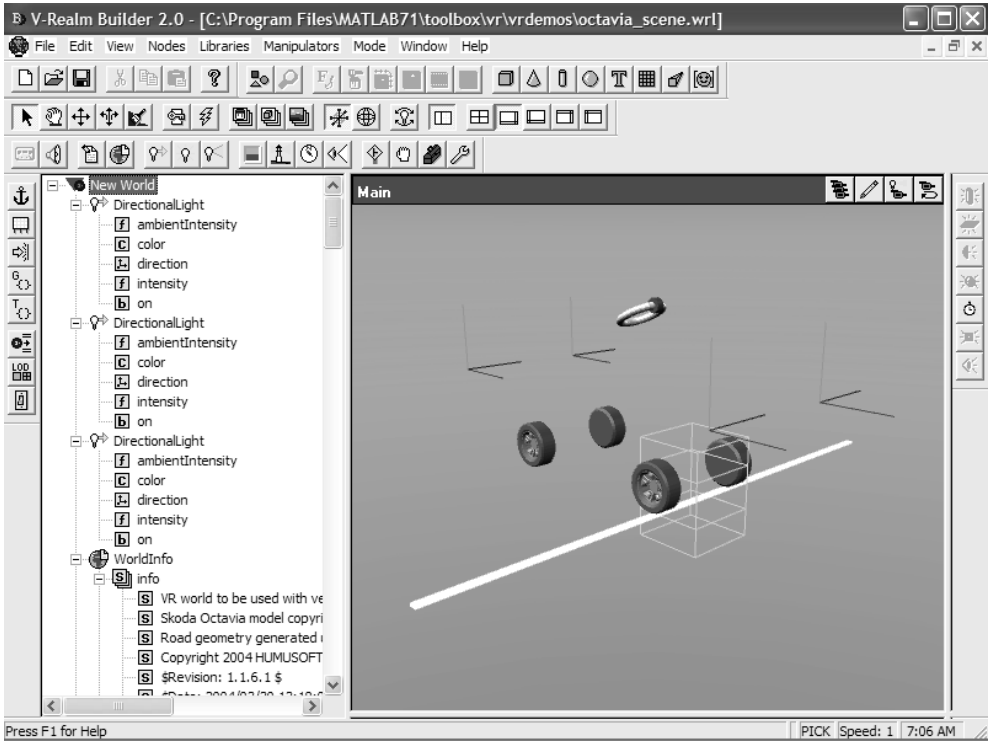


*Рис. 12.64. Окно редактора **VR** объектов*

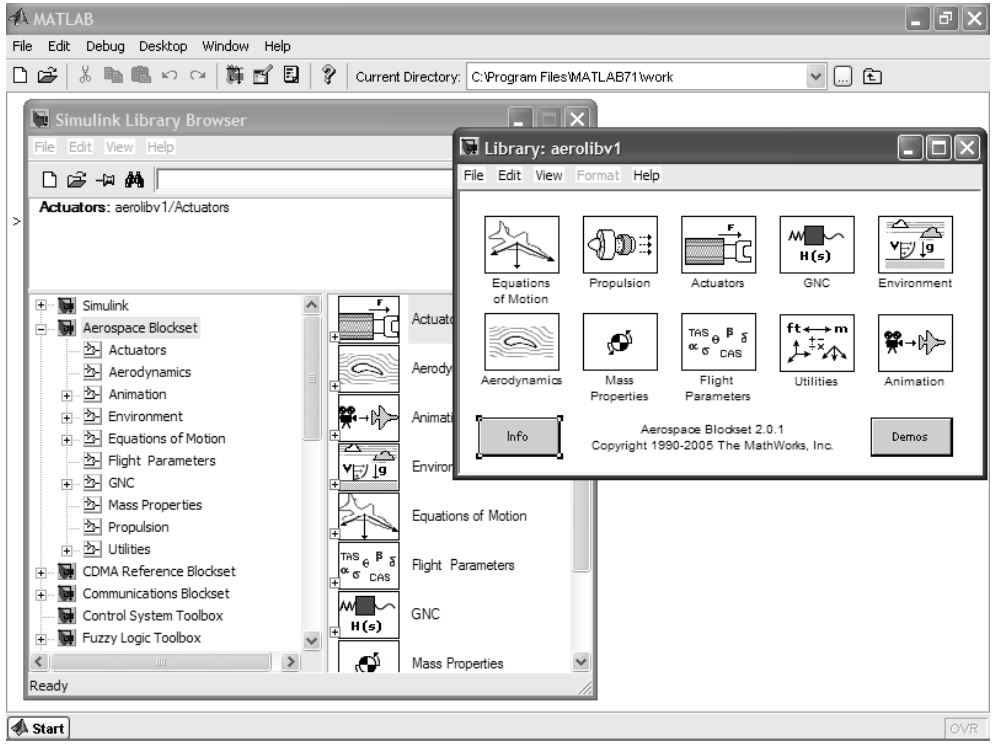


Рис. 12.65. Доступ к библиотеке пакета расширения Aerospace Blockset

- **Fin Deflection** – отклонение стабилизатора аппарата;
- **Center of Gravity** – центр тяжести аппарата;
- **Normal Acceleration η** – нормальное ускорение аппарата;
- **Incidence α** – угол атаки;
- **Body Rate q** – проекция вектора абсолютной угловой скорости аппарата.

12.8.3. Блоки задания уравнений движения 6DoF и 3DoF

К числу наиболее важных блоков расширения относятся блоки **Equations of Motions** для задания уравнений движения. Как видно по рис. 12.66 (левый нижний угол), в состав библиотеки этих блоков входят три раздела. В первом разделе **6DoF** находятся блоки, задающие движения в пространстве с шестью степенями свободы (перемещения вдоль осей координат объекта и повороты вокруг их). Всего таких блоков 15, три из них видны в окне, показанном на рис. 12.68.

Окна установки параметров этих блоков отличаются незначительно. В связи с этим приведем окно установки блока **6DoF** (Euler Angles), который осуществляет представление углового движения тела через углы Эйлера. Оно представлено на рис. 12.69.

Ниже представлены параметры, которые устанавливаются в этом окне:

- `Initial position in internal axes [Xe,Ye,Ze](m)` – начальное положение в инерциальной (земной) системе координат (отклонение центра масс *O* летательного аппарата от начала земной системы координат);
- `Initial velocity in body axex [u,v,w](m/s)` – проекции начальной скорости на оси тела в начальный момент времени;
- `Initial Euler oriaentation [roll,pitch,yaw](rad)` – начальная ориентация в углах Эйлера (углы крена, тангажа и рыскания);
- `Initial body rotation rates [p,q,r] (rad/s)` – проекции начальной угловой скорости летательного аппарата на оси его координатной системы;
- `Mass (kg)` – масса летательного аппарата;
- `Inertia matrix (kg.m2)` – матрица моментов инерции летательного аппарата относительно осей его координатной системы (размера 3×3).

Входные параметры блоков данной группы были отмечены выше и несколько отличаются по набору. Например, для первых пяти блоков это вектор текущих

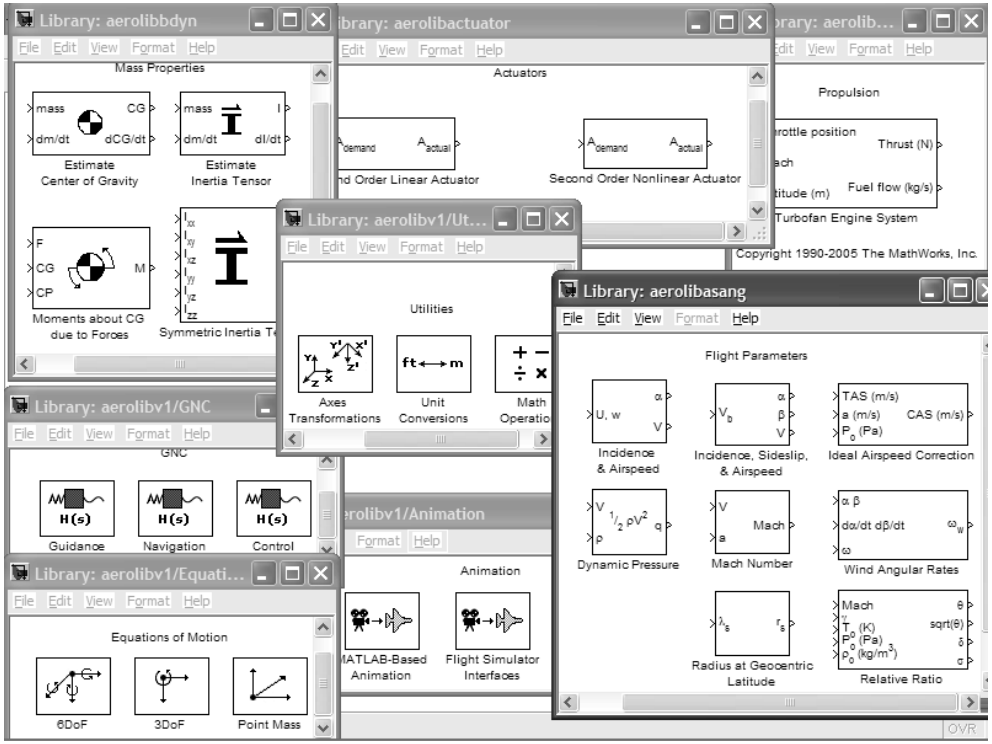


*Рис. 12.66. Окна с подразделами библиотеки пакета расширения **Aerospace Blockset***

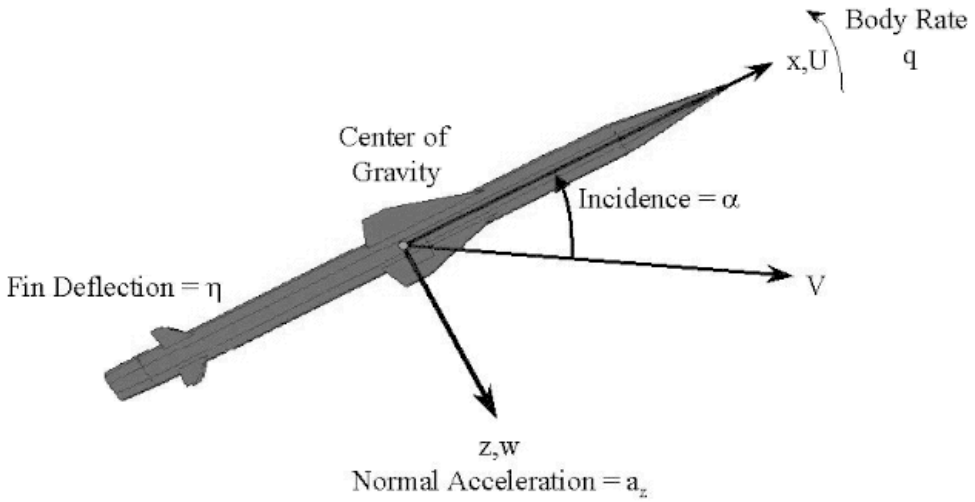


Рис. 12.67. Положение летательного аппарата в пространстве

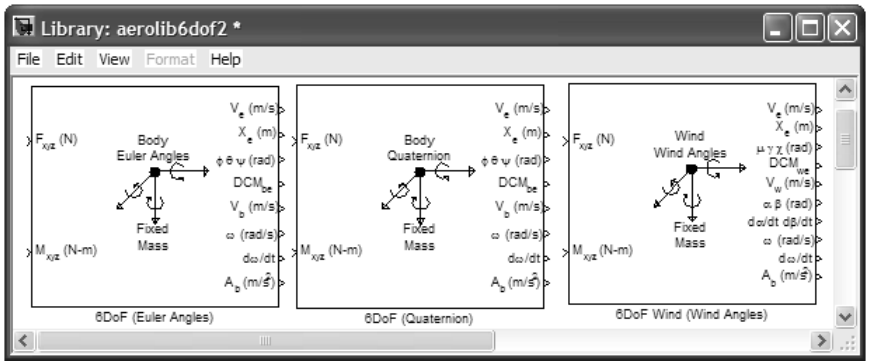


*Рис. 12.68. Окно с частью блоков раздела **6DoF***

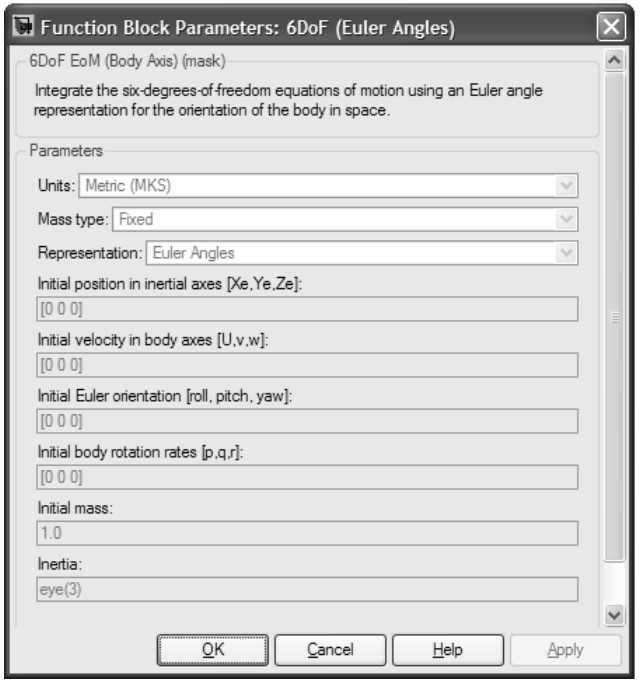


*Рис. 12.69. Окно установки параметров блока **6DoF** (Euler Angles)*

проекций на оси тела всех действующих на него внешних сил и вектор текущих моментов сил относительно осей тела. Выходные параметры блоков следующие:

- `Ve (m/s)` – вектор проекций текущего значения вектора скорости центра масс тела на оси инерциальной (земной) системы координат;

- `Xe (m)` – вектор текущих смещений центра масс тела относительно начала координат инерциальной (земной) системы координат;
- `Euler (rad)` – вектор текущих значений углов Эйлера (крена, тангажа и рыскания);
- `DCM` – текущее значение матрицы направляющих косинусов связанных осей относительно осей инерциальной (земной) системы координат;
- `Vb (m/s)` – вектор проекций текущего значения вектора скорости центра масс тела на оси системы его координат;
- `p, q, r (rad/s)` – вектор проекций текущей угловой скорости тела на оси его системы координат;
- `pdot, qdot, rdot (rad/s2)` – вектор производных от проекций текущей угловой скорости тела на оси его системы координат.

Блоки **3DoF** (рис. 12.70) служат для моделирования движения тел (летательных аппаратов) в одной плоскости с тремя степенями свободы. Обычно моделируется продольное движение в вертикальной плоскости.

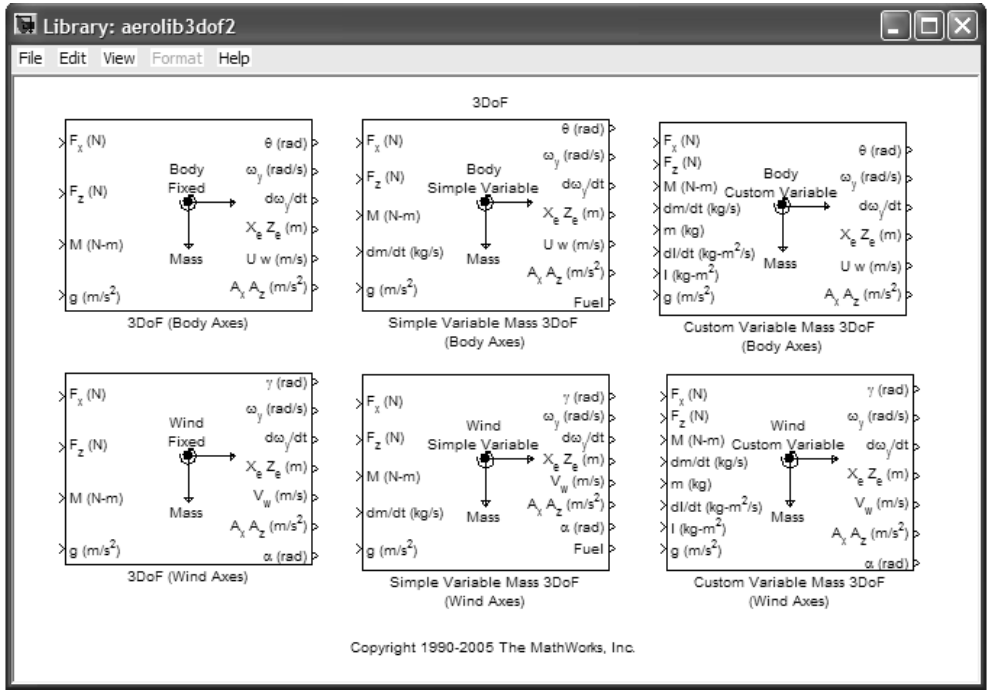


*Рис. 12.70. Окно библиотеки **3DoF***

Входные величины блоков достаточно очевидны из их наименований. Например, у первого блока это:

- `Fx (N)` – текущее значение силы (в ньютонах), действующей на тело, вдоль его продольной оси X_b;

- `Fz (N)` – текущее значение силы (в ньютонах), действующей на тело, вдоль его поперечной оси Z_b;
- `M (N-m)` – текущее значение момента сил (в ньютонах на метр), действующего на тело, вокруг его поперечной оси Z_b;
- `g (m/s2)` – значение ускорения свободного падения тела.

Выходные параметры блоков немного различаются (в основном по углам, определенным выше). Приведем список выходных параметров, характерных для всех блоков этой группы:

- `XeZe (m)` – текущие координаты центра масс тела в продольной плоскости в инерциальной системе координат (в метрах);
- `U, w (m/s)` – текущие значения проекций скорости тела на его продольную и поперечную оси его системы координат (в метрах в секунду);
- `Ax, Az (m/s2)` – текущие значения проекций ускорений тела на продольную и поперечную оси его системы координат (в метрах в секунду в квадрате).

Окна установки параметров этих блоков имеют обычный вид. В связи с этим ограничимся указанием параметров для первого блока этого раздела библиотеки (рис. 12.71):

- `Units` – выбор системы единиц измерения (выбор из списка);
- `Mass type` – тип масс (выбор из списка);

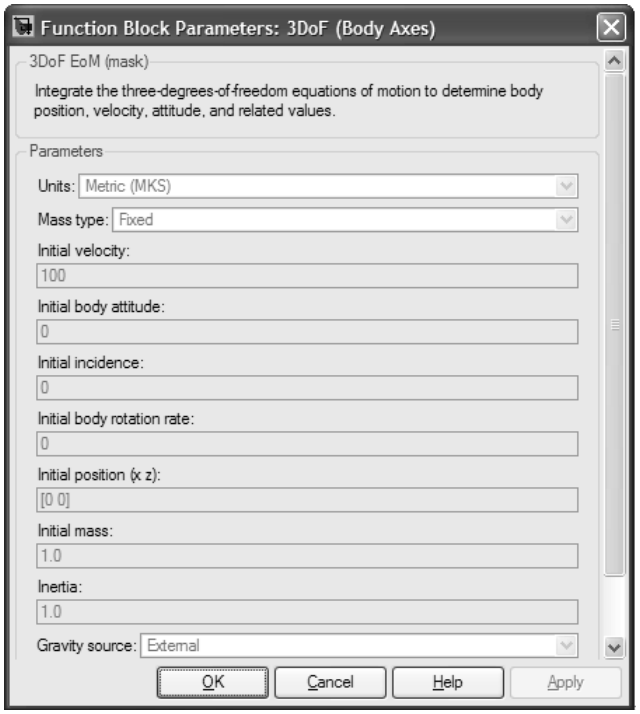


*Рис. 12.71. Окно параметров первого блока библиотеки **3DoF***

- `Initial velocity (m/s)` – начальная скорость;
- `Initial body altitude (rad)` – начальный угол атаки;
- `Initial incidence (rad)` – установка угла атаки;
- `Initial body rotation rates (rad/s)` – начальная угловая скорость тангажа;
- `Initial position (m)` – начальное положение центра масс;
- `Mass (kg)` – масса тела (летательного аппарата);
- `Inertia (kg.m2)` – момент инерции относительно поперечной оси;
- `Acceleration due to gravity (m/s2)` – ускорение свободного падения тела;
- `Gravity Source` – источник гравитации (выбор из списка).

В некоторых блоках раздела **3DoF** имеются дополнительные параметры:

- `Initial flight path angle` – начальный угол пути полета;
- `Empty mass` – чистая масса;
- `Full mass` – полная масса;
- `Empty inertia` – чистый момент инерции;
- `Full inertia` – полный момент инерции.

Блоки **6DoF** и **3DoF** обеспечивают решение дифференциальных уравнений, описывающих движение тел (летательных аппаратов) в пространстве и в плоскости. Для этого в их состав входят средства интегрирования таких дифференциальных уравнений. Обеспечение работы этих блоков требует формирования текущих значений сил и моментов сил следующих трех групп:

- силы и моменты, действующие на летательный аппарат со стороны маршевого двигателя;
- силы и моменты сил, действующие со стороны окружающей среды (сила тяжести и силы сопротивления воздуха);
- силы и моменты сил, возникающие вследствие работы исполнительных механизмов аппаратов (рули, элероны, элевоны, закрылки и т. д.).

12.8.4. Блок системы турбовентиляторного двигателя

Для моделирования маршевого двигателя служит блок турбовинтового двигателя **Propulsion** – рис. 12.72.

Этот блок имеет следующие входы:

- **Throttle position** – текущее положение регулирующего дросселя;
- **Mach** – текущее значение числа Маха;
- **Altitude (m)** – текущее значение высоты полета (в метрах).

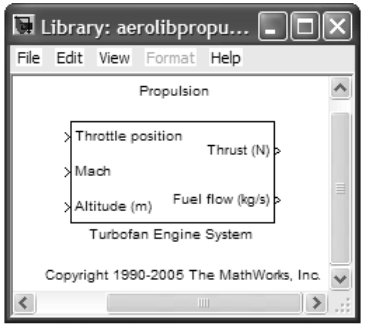


*Рис. 12.72. Окно блока **Propulsion***

Выходы блока следующие:

- **Thrust (N)** – сила тяги (в Ньютонах);
- **Fuel flow (kg/s)** – расход горючего в единицу времени (кг/с).

Окно установки параметров этого блока показано на рис. 12.73.

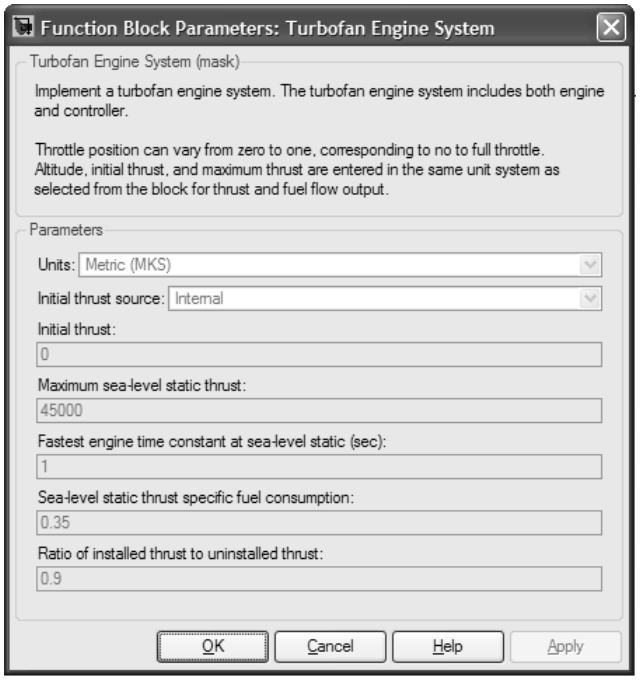


*Рис. 12.73. Окно установки параметров блока **Propulsion***

В окне параметров блока устанавливаются следующие параметры:

- `Units` – система единиц измерения (выбор из списка);
- `Initial thrust source` – источник начального значения силы тяги (внутренний или внешний);
- `Initial thrust` – начальное значение силы тяги;
- `Maximum sea level static thrust` – максимальное значение силы тяги на уровне океана;
- `Faster engine time constant at sea kevel static (sec)` – постоянная времени двигателя (на уровне океана);
- `Sea level static fuel consumption` – удельный расход топлива двигателя на уровне океана;
- `Ratio of installed thrust to uninstalled thrust` – отношение установленной силы тяги к неустановленной.

12.8.5. Блоки учета влияния среды раздела Environment

Для учета влияния окружающей среды на полет летального аппарата служат блоки раздела **Environment** – рис. 12.74. Всего здесь имеются три раздела: **Atmosphere** – учета влияния атмосферы, **Gravity** – учета влияния гравитации и **Wind** – учета влияния ветра.

Раздел **Atmosphere** содержит шесть блоков, представленных на рис. 12.75. Первые пять блоков содержат модели атмосферы на высоте полета h (m), заданной в метрах. Три блока описывают стандартные модели атмосферы ISA, Lapse и COESA. Имеются также две нестандартные модели и блок **Pressure Altitude** для вычисления атмосферного давления на заданной высоте.

*Рис. 12.74. Окно блоков раздела библиотеки **Environment***

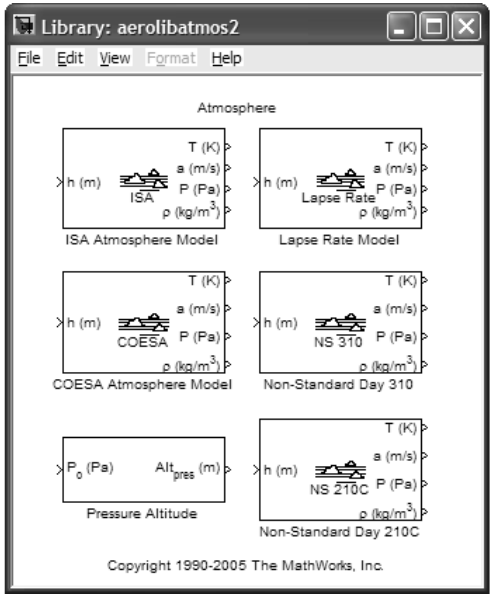


*Рис. 12.75. Окно блоков раздела **Atmosphere** библиотеки **Environment***

Первые пять блоков по заданной высоте h (в метрах) вычисляют следующие параметры атмосферы:

- `Temperature (K)` – температура в градусах Кельвина;
- `Speed of sound (m/s)` – скорость звука в м/с;
- `Air Pressure (N/m2)` – давление воздуха в ньютонах на метр в квадрате;
- `Air Density (kg/m3)` – плотность воздуха в кг/м^3.

Окна установки параметров первых пяти блоков просты, и разобраться в них несложно. На рис. 12.76 показано окно настройки блока модели атмосферы ISA. Оно вообще не имеет параметров. Этот блок рассчитывает параметры атмосферы

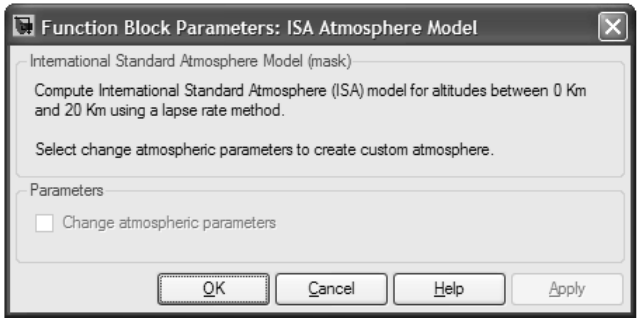


*Рис. 12.76. Окно установки параметров блока модели **ISA** атмосферы*

по стандартной интернациональной методике до высот в 20 000 м. Блок **COESA** использует американскую методику расчета параметров атмосферы до высот 84 852 м. С параметрами остальных блоков и их возможностями заинтересованный читатель может ознакомиться самостоятельно.

В первой версии пакета **Aerospace Blockset** был единственный блок, вычисляющий гравитационную постоянную на текущей высоте и географической широте. Теперь в разделе **Gravity** три блока – рис. 12.77. Это уже названный блок **VGS84 Gravity Model** и еще два блока, описывающих модели магнитного поля Земли. В связи с редким применением этих блоков подробное описание их не приводится.

В блоке установки параметров блока задаются система физических единиц и тип сообщения при выходе параметра за допустимые пределы. В остальных блоках задается также дата (день, месяц и год).

Раздел **Wind** содержит 12 блоков, моделирующих влияние ветра при различных его моделях. Окно с этими блоками представлено на рис. 12.78. Назначение небольшого числа входных и выходных параметров этих блоков очевидно из названия параметров, а название блоков явно указывает на тип модели. В большинстве случаев модели названы именами их создателей. Параметры блоков до-

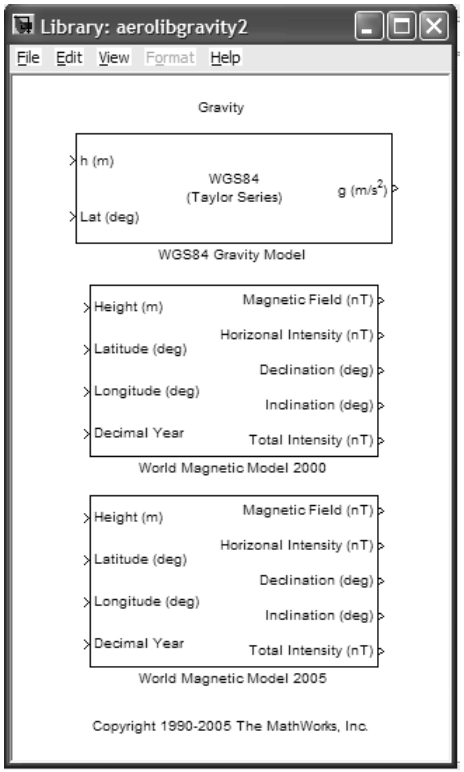


*Рис. 12.77. Окно блоков раздела **Gravity***

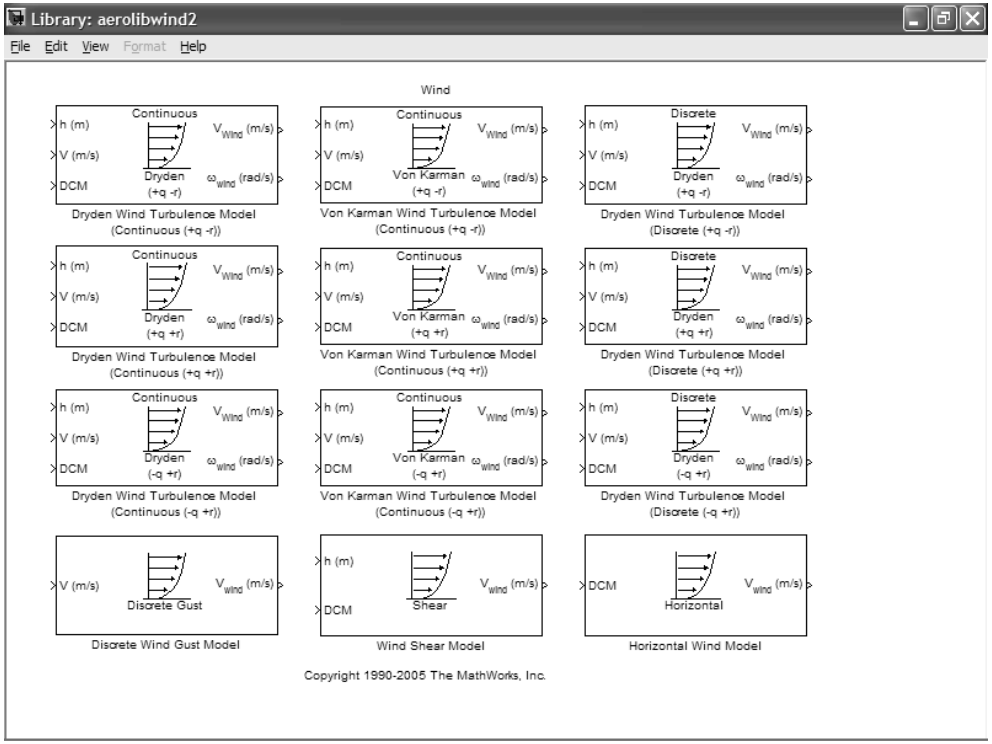


*Рис. 12.78. Окно блоков раздела **Wind***

статочно разнообразны, но просты. Заинтересованный пользователь пакетом **Aerospace Blockset** может легко разобраться с их установкой.

12.8.6. Блоки исполнительных механизмов – Actuators

В разделе **Actuators** (см. окно на рис. 12.66 сверху и в центре) имеются два блока, моделирующих работу силового привода рулевых органов управления:

- **Second Order Linear Actuator** – линейный силовой привод второго порядка;
- **Second Order Nonlinear Actuator** – нелинейный силовой привод второго порядка.

Оба блока имеют один вход `Ac_dem ()` для указания текущего положения исполнительного элемента и одни выход `Ac_ac` с требуемым положением исполнительного элемента.

В окне параметров блока **Second Order Linear Actuator** можно установить следующие параметры:

- `Natural frequency` – частота собственных колебаний;

- `Damping ratio` – коэффициент затухания;
- `Initial position` – начальное положение регулирующего органа.

В окне параметров блока **Second Order Nonlinear Actuator**, кроме этих параметров, можно установить еще три параметра:

- `Maximum deflection` – максимальное отклонение регулирующего органа;
- `Minimum deflection` – минимальное отклонение регулирующего органа;
- `Maximum rate` – максимальная скорость отклонения регулирующего органа.

Таким образом, вводятся ограничения, которые и отличают работу нелинейного блока от линейного. Оба блока моделируют типовую инерционность регулирующих органов, характерную для систем второго порядка. В зависимости от установленного коэффициента затухания изменение отклонения может носить как апериодический, так и колебательный затухающий характер.

12.8.7. Блоки регуляторов управления движением – GNC

Блоки регуляторов управления движением в разделе библиотеки **GNC** (см. рис. 12.66 окно слева в центре) содержатся в трех подразделах:

- **Guidance** – блок измерения расстояния между воздушными судами;
- **Navigation** – блоки навигации;
- **Control** – блоки контроля.

Полный состав этих блоков представлен на рис. 12.79.

Раздел **Guidance** содержит единственный блок **Calculate Range**, который вычисляет расстояние между двумя воздушными судами (самолетами), заданными их позициями. Блок установки параметров у этого блока чисто информационный и каких-либо установок не имеет.

Раздел **Navigation** содержит три блока:

- **Three-axis Accelerometer** – блок измерителя ускорения по трем осям;
- **Three-axis Gyroscope** – блок гироскопа по трем осям;
- **Three Axis Internal Measurement Unit** – встроенный универсальный измеритель.

Входные и выходные параметры этих блоков вполне очевидны и уже описывались. В блоках установки параметров блоков устанавливаются также уже описанные ранее параметры, например система единиц измерения, местоположение блока, частота собственных колебаний, коэффициент демпфирования и др.

Самое большое число блоков находится в разделе **Control**. Все они видны в основном окне **Control**, показанном на рис. 12.79. Эти блоки делятся на пять характерных групп:

- блоки одномерного 1D-контроля (4 блока);
- блоки двумерного 2D-контроля (4 блока);
- блоки трехмерного 3D-контроля (3 блока);
- вспомогательные блоки контроля (2 блока);

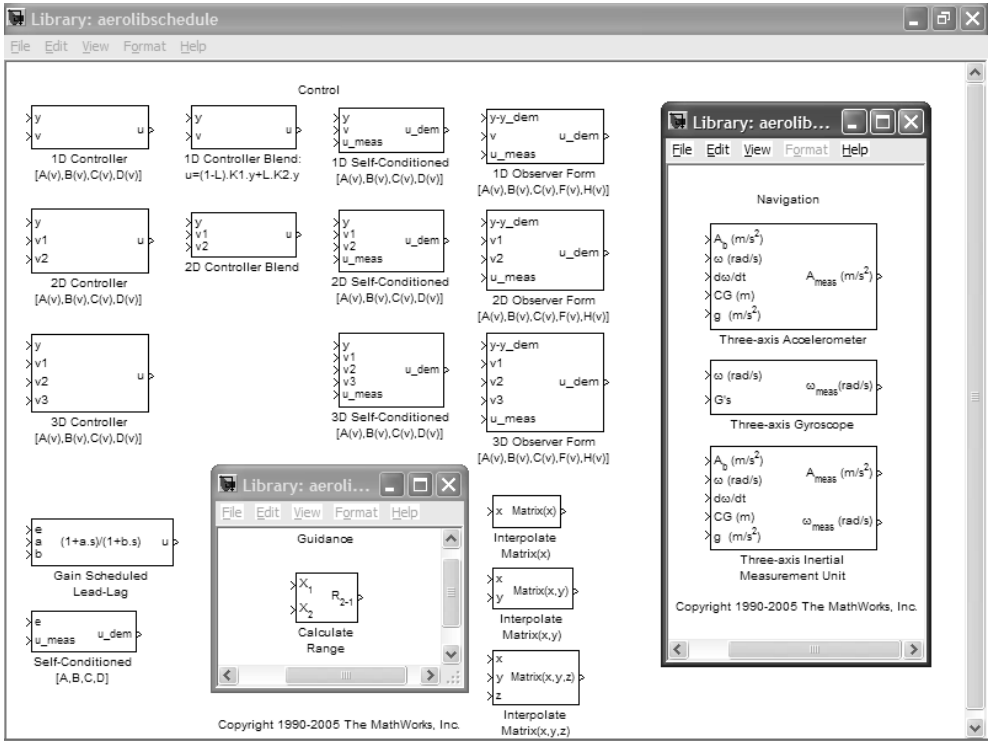


*Рис. 12.79. Окна блоков раздела **GNC***

- вспомогательные блоки интерполяции матричных данных (3 блока для 1D-, 2D- и 3D-матриц).

Все блоки, за исключением трех блоков интерполяции матричных операций, имеют один выходной параметр *u* – требуемый угол поворота исполнительного механизма. Входными являются данные (матрицы), которые вырабатывают измерительные приборы, устанавливаемые на борту летательных аппаратов. Данные берутся из рабочего пространства системы MATLAB и соответствуют матричной форме представления его (модель A,B,C,D). Кроме этих матриц, задаются три переменные (v1, v2 и v3) точек останова и начальное значение x – см. рис. 12.80.

12.8.8. Блоки свойства масс – Mass Properties

В подраздел библиотеки **Mass Properties** входят четыре блока – рис. 12.66 окно слева и сверху. Это следующие блоки:

- **Estimate Center of Gravity** – оценка центра гравитации с применением линейной интерполяции;
- **Estimate Inertia Tensor** – оценка тензора инерции;

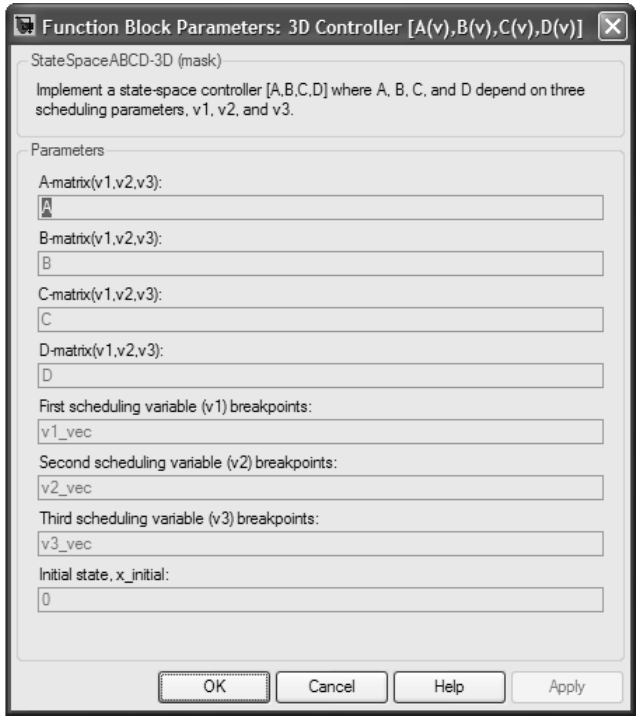


*Рис. 12.80. Окно установки параметров блока **3-D Controller***

- **Moments of CG due to Forces** – вычисление момента сил;
- **Symmetric Inertia Tensor** – вычисление симметричного тензора инерции.

Для двух первых блоков входными являются масса mass и производная dm/dt. Выходные параметры первого блока – это центр гравитации CG и его изменение во времени dCG, для второго это тензор инерции I и его приращение во времени dI/dt. В окнах установки параметров этих блоков задаются полная и чистая массы, а также полный и чистый центры гравитации или полная и чистая матрица инерции (для блока **Estimate Inertia Tensor**. Другие два блока имеют информационные окна установки параметров).

12.8.9. Блоки вычисления параметров полета – Flight Parameters

Раздел библиотеки **Flight Parameters** (рис. 12.66 справа снизу) содержит восемь блоков:

- **Incidence & Airspeed** – вычисление угла атаки и скорости;
- **Incidence, Slideslip & Airspeed** – вычисление угла атаки и скорости;
- **Ideal Airspeed Correction** – коррекция идеального значения скорости;

- **Dynamic Pressure** – вычисление динамического давления;
- **Mach Number** – вычисление числа Маха;
- **Wing Angular Rates** – вычисление угловых пределов ветра;
- **Radius at Geocentric Latitude** – вычисление радиуса в геоцентрической широте;
- **Relative Ratio** – вычисление относительных температуры, квадратного корня от температуры, давления и плотности воздуха.

У всех этих блоков установочных параметров нет и окна установки параметров являются информационными. Исключение составляет блок **Ideal Airspeed Correction**, окно установки параметров которого представлено на рис. 12.81. Этот блок вычисляет эквивалентную EAS, калиброванную CAS или верную TAS скорость (нужный вид скорости проставляется у выходного порта блока). В окне параметров задаются система физических единиц измерения, тип скорости на входе и выходе блока и тип реакции при выходе входного параметра за допустимые пределы. Входные параметры блоков были описаны выше при описании других блоков.

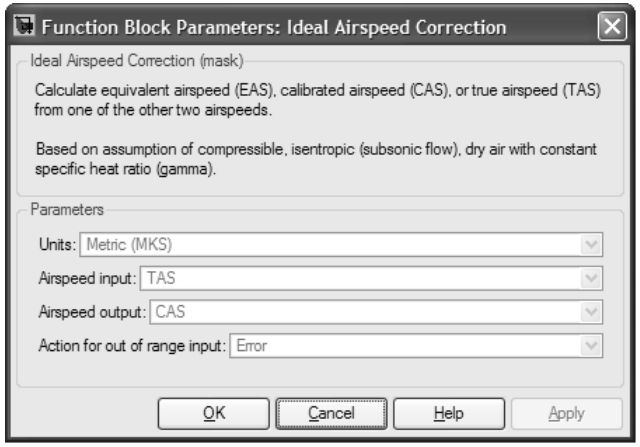


Рис. 12.81. Окно установки параметров блока ***Ideal Airspeed Correction***

12.8.10. Блок аэродинамики – Aerodynamic

Раздел библиотеки **Aerodynamic** (рис. 12.82) содержит единственный блок **Aerodynamic Forces and Moments**. Блок вычисляет аэродинамические силы и моменты, относящиеся к центру гравитации по заданным на входе коэффициентам аэродинамики, динамическому давлению CP и центру координатной системы CG.

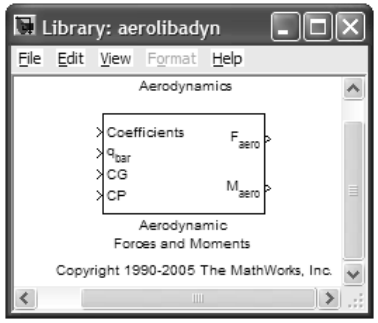


Рис. 12.82. Окно раздела библиотеки ***Aerodynamic***

Окно установки параметров этого блока позволяет задать справочную площадь, размах крыла и длину летающего объекта.

12.8.11. Блоки раздела утилит – Utility

Блоки раздела утилит библиотеки **Utility** (рис. 12.83 и 12.84) делятся на три категории:

- **Axes Transformation** – преобразования осей;
- **Math Operations** – математические операции;
- **Unit Conversion** – преобразования физических величин.

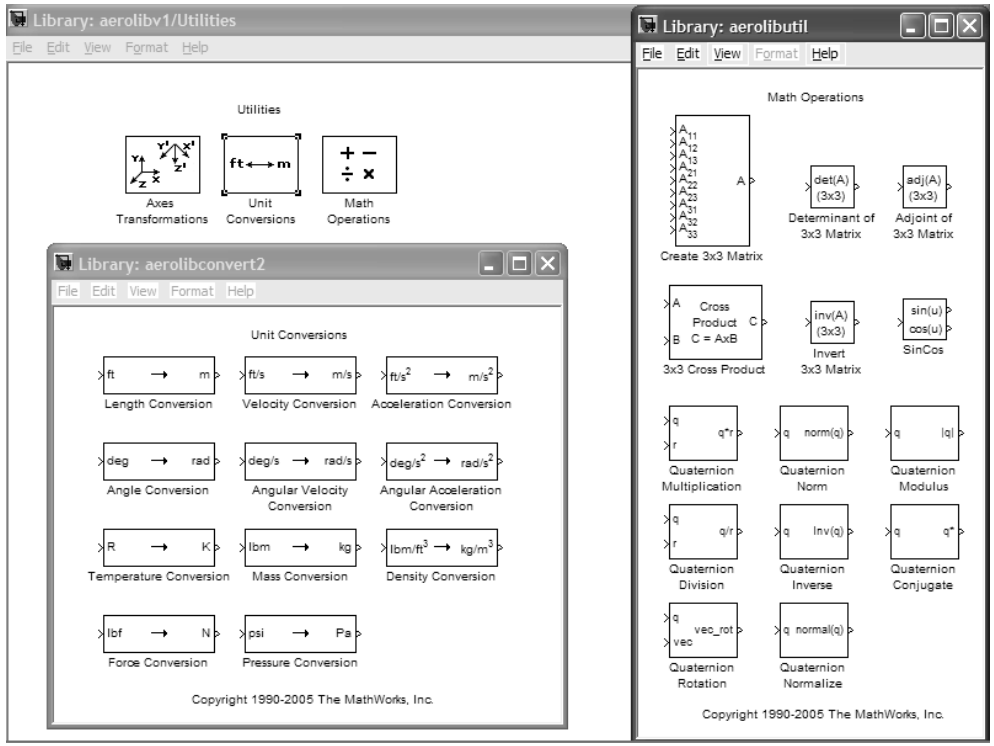


Рис. 12.83. Окна разделов ***Unit Conversion*** *и* ***Math Operations***

Раздел **Unit Conversion** содержит 11 простых блоков, назначение которых вполне очевидно – каждый блок осуществляет преобразование, указанное его именем. Например, первый блок **Length Conversion** преобразует длину, второй блок **Velocity Conversion** – скорость и т. д. Все блоки имеют однотипные окна

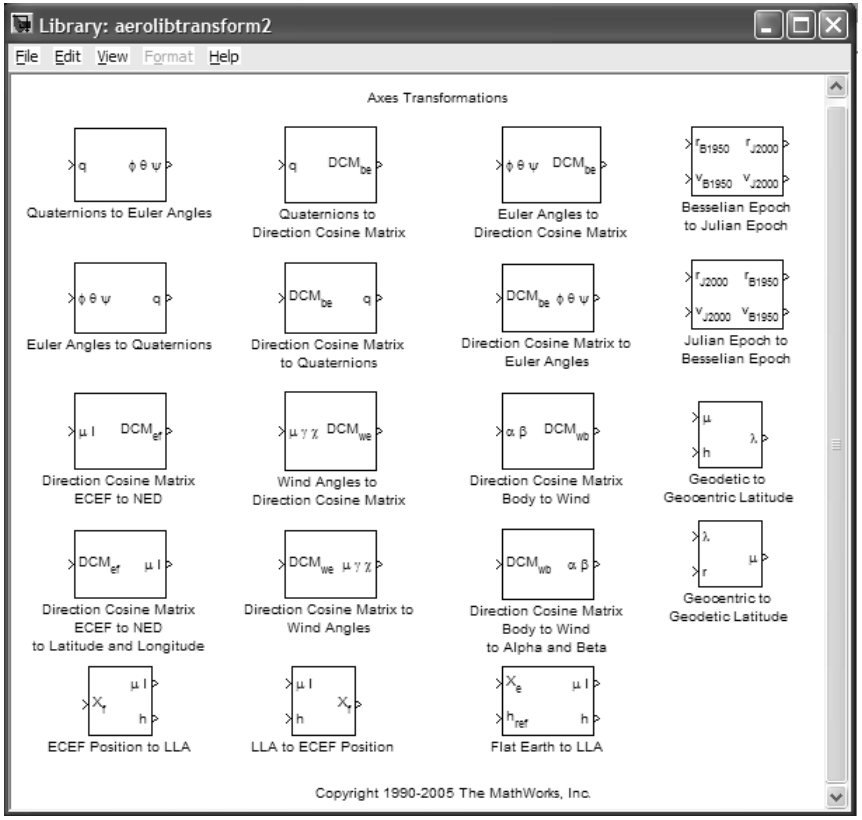


*Рис. 12.84. Окно раздела **Axes Transformation***

установки параметров, имеющие всего две установки – системы единиц измерения входной величины и выходной величины.

Блок **Math Operation** содержит 14 блоков для выполнения математических операций, часто используемых при моделировании аэрокосмических аппаратов. Все эти блоки имеют очевидное из их названия назначение и в особом описании не нуждаются. Интересно, что все они – чисто информационные окна установки параметров.

Самый большой раздел **Axes Transformation** содержит 19 блоков – рис. 12.84. Их назначение также вполне очевидно. Большинство блоков этой группы имеют чисто информационные окна установки параметров. Те же блоки, которые имеют окна с устанавливаемыми параметрами, имеют таковые, уже упомянутые выше.

12.8.12. Блоки анимации – Animation

Для наглядного представления результатов моделирования полета аэрокосмических аппаратов в **Aerospace Blockset** введены средства анимации. Они представлены в трех подразделах (рис. 12.85):

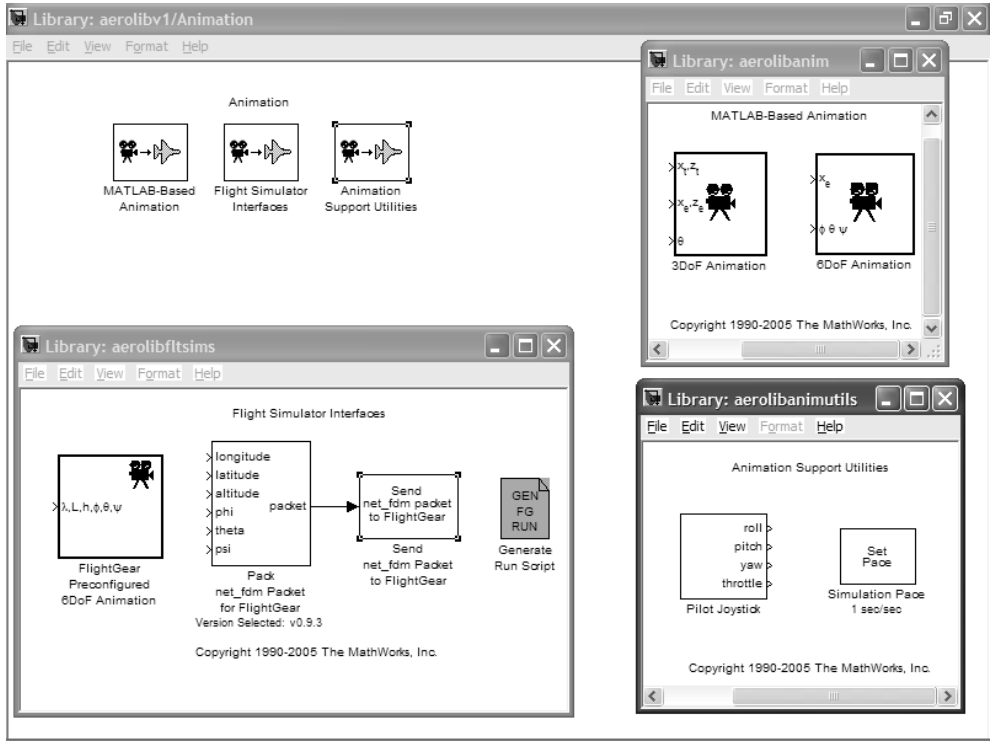


*Рис. 12.85. Окна раздела **Animation***

- **MATLAB –Based Animation** – блоки базовой анимации MATLAB;
- **Flight Simulator Interface** – интерфейс имитатора полета;
- **Animation Support Utilities** – утилиты поддержки анимации.

В первый подраздел MATLAB – **Based Animation** – входят два блока для анимационного построения движения аппаратов с тремя **3DoF** и шестью **6DoF** степенями свободы. Анимация осуществляется средствами базовой системы MATLAB.

Блок **3DoF** создает анимацию летательного аппарата в пространстве по заданным *XZ* позициям и высотой над уровнем моря. В окне установки параметров блока устанавливаются: пределы значений величин по трем осям, время обновления картинки, размер аппарата, позиция камеры просмотра и угол обзора.

Блок **6DoF** создает анимацию летательного аппарата в пространстве по заданным *XYZ* позициям и углам Эйлера. В окне установки параметров блока устанавливаются те же параметры, что и у блока **3DoF**. Дополнительно задается статическая позиция аппарата (объекта).

Во второй раздел **Flight Simulator Interface** входят четыре блока. Блок **FlightGearQuick6DoFAnimation** дает улучшенную анимацию для объектов с шестью степенями свободы. Его входные величины задаются числами с двойной точностью. В блоке установки параметров задаются версия FlightGear, IP-адресс, порт и время Simple time.

Блок-маска **FlightGearPackNetFdm** является компактной сетевой версией симулятора полетов. Он подключается к блоку **Send net_fdm Packet to FlightGear**. В блоке параметров устанавливаются версия FlightGear и опции показа тех или иных параметров.

Последний блок-маска **Generate FlightGear Run Script** не имеет входных и выходных параметров. Но окно установки его параметров позволяет задать множество параметров анимации. Оно показано на рис. 12.86.

Рис. 12.86. Окно установки параметров блока ***Generate FlightGear Run Script***

Возможности средств анимации можно более детально изучить по многочисленным примерам применения пакета **Aerospace Blockset** в разделе **Demos** справки.

12.9. Примеры применения пакета расширения Aerospace Blockset

12.9.1. Доступ к демонстрационным примерам

Прилагаемые к пакету **Aerospace Blockset** демонстрационные примеры на первых порах избавляют пользователя от кропотливой работы по составлению диаграмм моделей летательных аппаратов и подбору их параметров. Эти примеры подготовлены профессионалами – разработчиками пакета и готовы к немедленному пуску и изучению. Однако надо помнить, что такие примеры создают обманчивое впечатление о простоте моделирования в такой сложной области науки и техники, как аэрокосмические аппараты. Поэтому никогда не стоит забывать, что эти примеры носят учебный характер и после их изучения придется потратить немало времени и сил для решения пользователем своих специфических задач в этой области.

Демонстрационные примеры находятся в папке aerodemos, которая, в свою очередь, расположена в папке aeroblks раздела файловой системы toolbox. В папке aerodemos находятся не только файлы примеров, но и файлы блоков-масок, нужные для сложных примеров с субблоками. Помимо запуска с панели **Demos** справки (рис. 12.87), примеры можно запускать и из командной строки MATLAB.

12.9.2. Простейшие примеры моделирования линейного силового привода

Диаграммы моделей в пакете расширения **Aerospace Blockset** строятся совершенно аналогично тому, как это делается в основном пакете расширения системы MATLAB – Simulink. Более того, в отличие от пакета **SimMexhanics**, особых ограничений на применение блоков Simulink в диаграммах **Aerospace Blockset** нет. Покажем это на простейшем примере с названием aeroblktutorial. Введя название этого примера в командной строке MATLAB и нажав клавишу **ENTER**, можно наблюдать появление окна примера, показанного на рис. 12.88.

Этот пример настолько прост, что читатель может подготовить его самостоятельно, найдя нужные блоки в библиотеках Simulink и пакета **Aerospace Blockset** и обеспечив их соединение. В данном примере исследуется блок **Second Order Linear Actuator** – линейный силовой привод второго порядка. Для возбуждения блока используется блок генерации синусоидального сигнала **Sinus Wave**, а для контроля реакции блока на это воздействие использован обычный виртуальный осциллограф Scope.

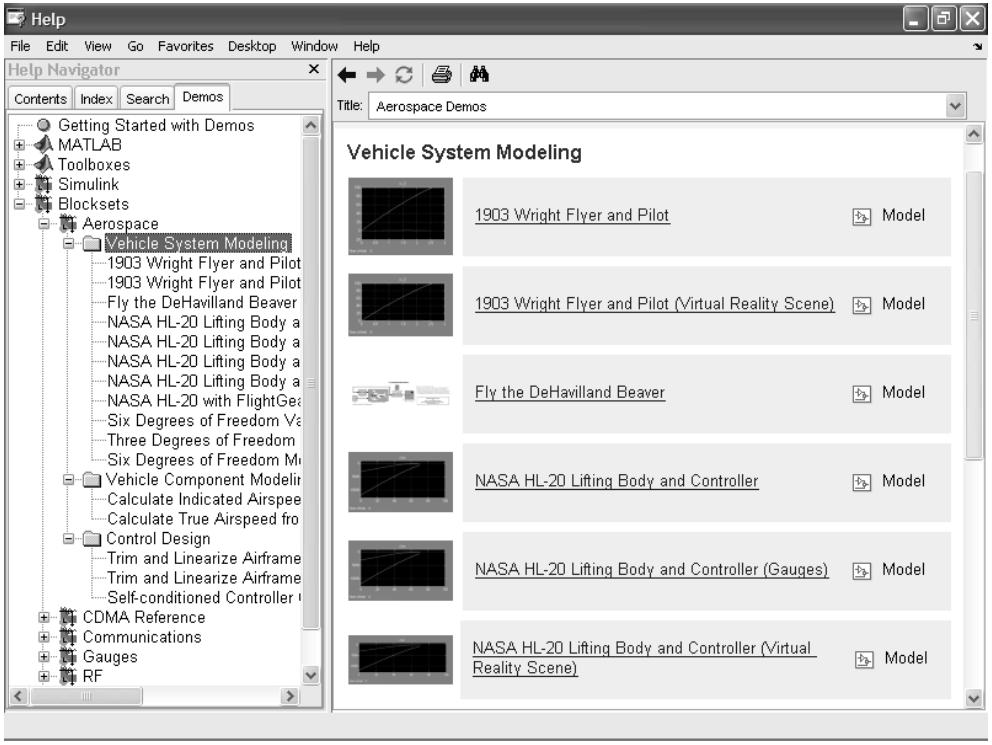


*Рис. 12.87. Доступ к демонстрационным примерам пакета с вкладки **Demos** справки*

На рис. 12.88 представлены окно задания параметров блока привода и осциллограммы сигналов на входе и выходе блока. Они практически совпадают, поскольку частота синусоидальных колебаний на входе блока намного меньше частоты собственных колебаний привода.

А теперь выполним следующую модификацию этого примера (рис. 12.89):

- заменим блок синусоидального сигнала на блок формирования прямоугольных импульсов **Pulse Generator** расширения Simulink;
- в окне параметров этого блока зададим период, равный 0,2 с;
- в окне параметров блока привода зададим коэффициент демпфирования, равный 0,1, что соответствует колебательной реакции блока второго порядка;
- установим время моделирования равным 0,5 с.

После пуска измененного примера можно наблюдать, насколько сильно изменилась работа блока привода. Его выходной сигнал стал довольно сложным и характерным для реакции колебательных систем на импульсное воздействие, в частности отчетливо видны затухающие и вначале довольно сильные колебания сигнала.

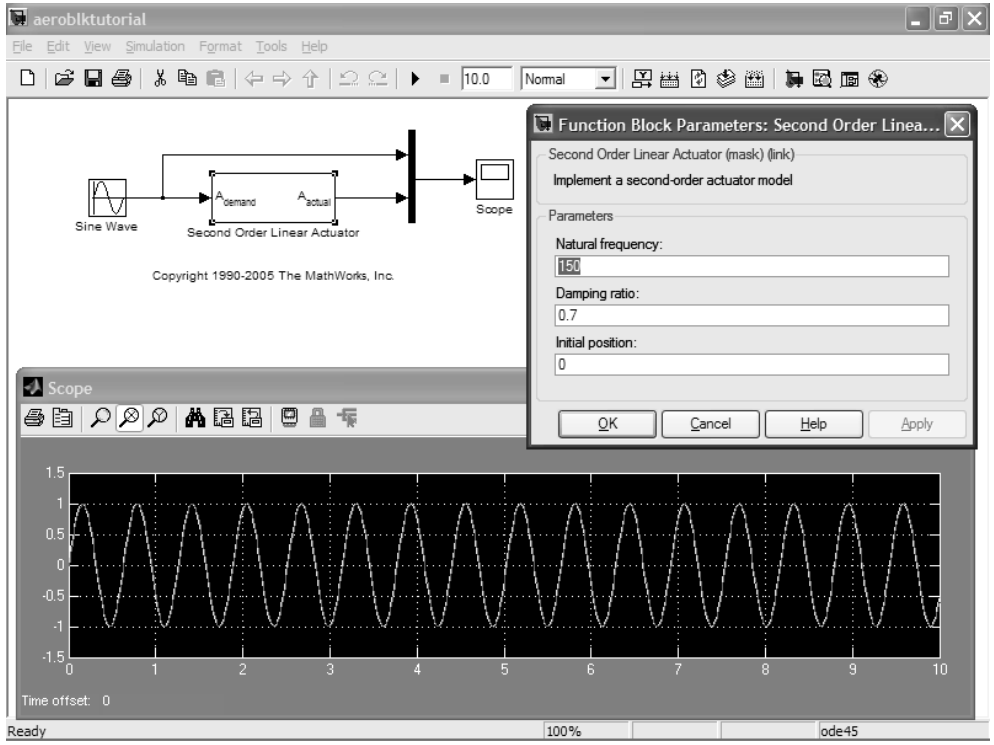


Рис. 12.88. Окно примера aeroblktutorial

12.9.3. Пример анимации при шести степенях свободы полета ракеты

Простой пример, иллюстрирующий возможности анимации на примере полета ракеты при шести степенях ее свободы, дан в файле aeroblk_vmm. Диаграмма модели и входящих в нее субблоков представлена на рис. 12.90. Этот пример наглядно демонстрирует применение субблоков и некоторых типовых блоков пакета расширения **Aerospace Blockset**. На этом рисунке представлено также изменение угла Эйлера и позиции центра масс аппарата в ходе его движения.

На самом деле анимация в данном примере сводится не к построению полета ракеты, она в окне выглядит неподвижной (рис. 12.91). Строится линия, соединяющая центр масс ракеты (точнее, ее каркаса) с центром Земли. При полете ракеты эта линия поворачивается, что и видно на рис. 12.91 в ходе моделирования. Кроме того, на рис. 12.91 представлен весь графический аппарат анимации, включая редактор графики, редактор свойств фигуры, палитры с векторами и матрицами, редактор свойств и т. д.

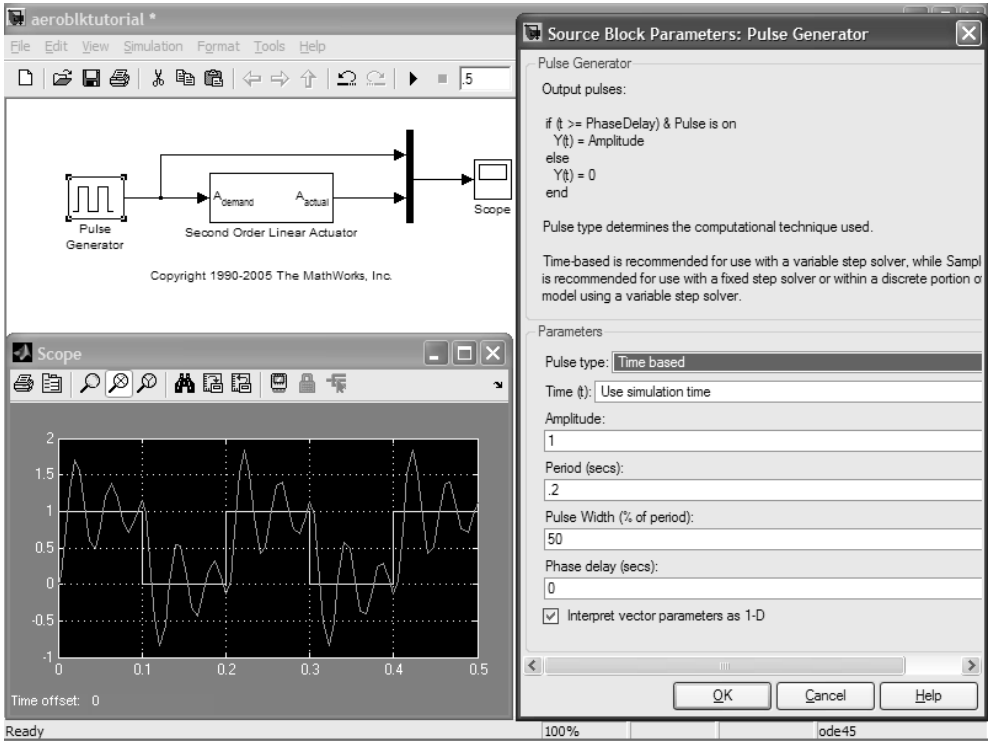


Рис. 12.89. Окно измененного примера aeroblktutorial

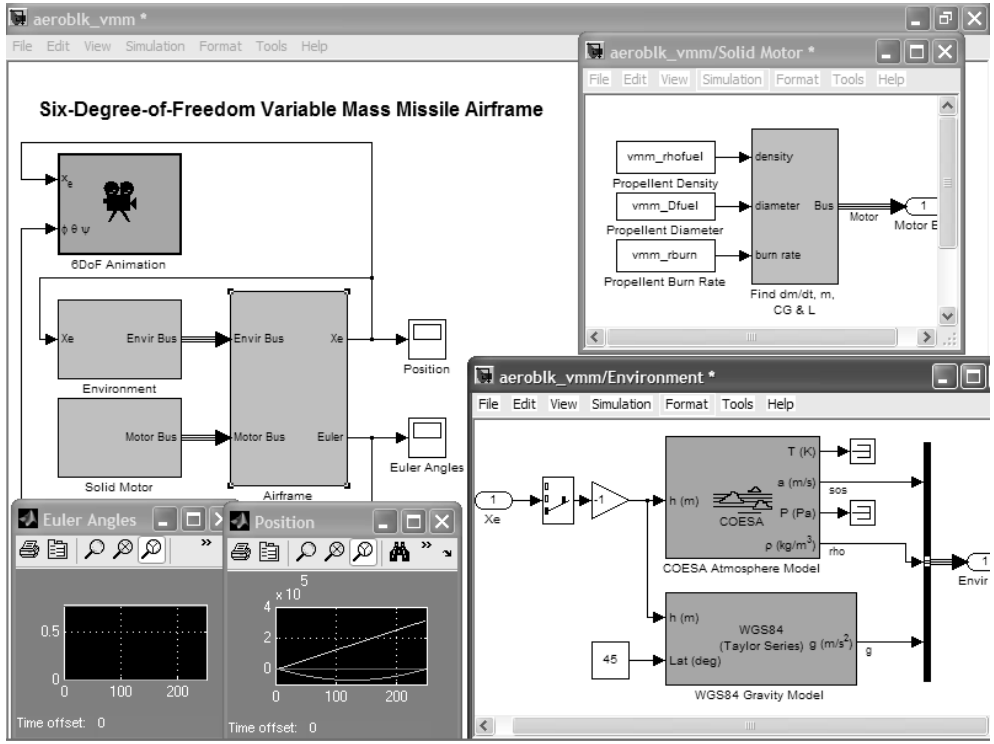


Рис. 12.90. Окно диаграммы примера aeroblk_vmm

12.9.4. Пример анимации при трех степенях свободы полета ракеты

Ограничение числа степеней свободы позволяет упростить решение задачи на анимацию движения летательного аппарата. В справке по пакету **Aerospace Blockset** описание средств пакета начинается с показательного примера aeroblk_guidance. Его диаграмма дана на рис. 12.92.

Вы можете с помощью редактора файлов MATLAB открыть файл Simulink-модели данного примера и убедиться в том, насколько он сложен даже для такого простого примера. Этот файл содержит более 6000 строк программного текста! Такова плата за визуально-ориентированное программирование для подготовки Simulink-модели. Можно просмотреть также MATLAB-файл с тем же именем, который используется для управления моделью и организации графического вывода результатов моделирования. Этот файл содержит всего около 40 строк кода.

Результаты моделирования по данной диаграмме представлены на рис. 12.93. Здесь даны часть осциллограмм, иллюстрирующих работу модели, и окно анимации, показывающее движение ракеты при трех степенях ее свободы.

12.9.5. Моделирование полета самолета – «этажерки»

В наши времена реактивной авиации кажется невероятным, что первые самолеты напоминали почти прозрачные этажерки. Красочный пример моделирования полета такого аппарата, созданного еще в 1903 г. (более 100 лет тому назад), представлен в списке демонстрационных примеров (файл aeroblk_wf_3dof). Его диаграмма с наглядным обозначением блоков показана на рис. 12.94.

Оставив за читателем удовольствие просмотреть все особенности моделирования для этого интересного примера, приведем лишь окно виртуального проигрывателя, демонстрирующего анимацию – полет «этажерки» (рис. 12.95).

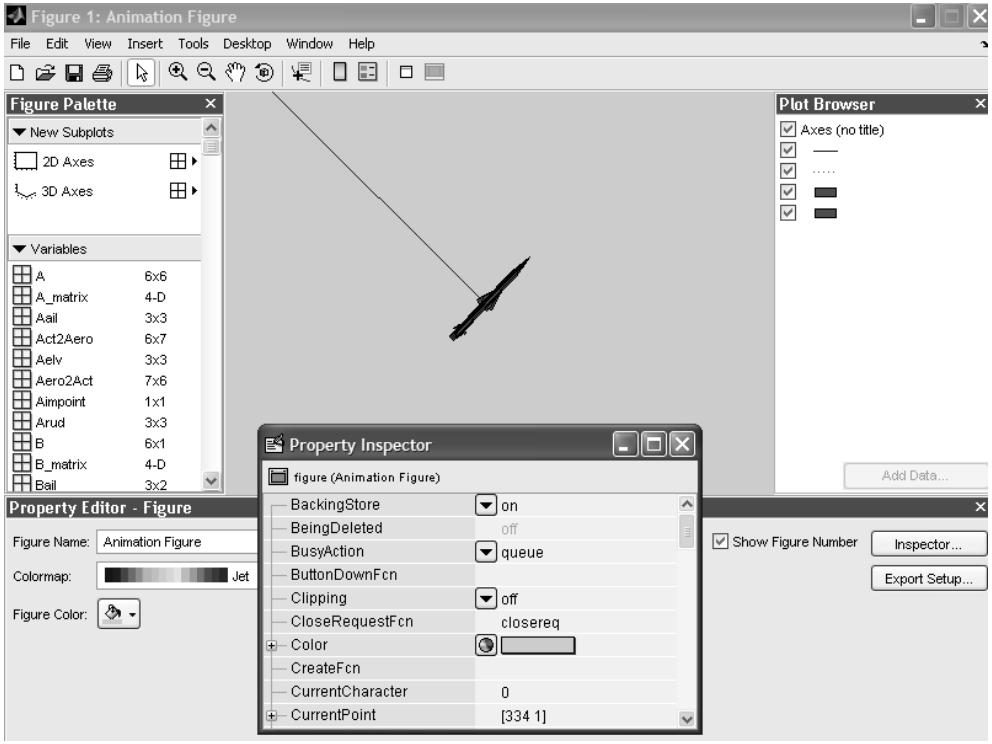


Рис. 12.91. Средства графической визуализации и анимации примера aeroblk_vmm

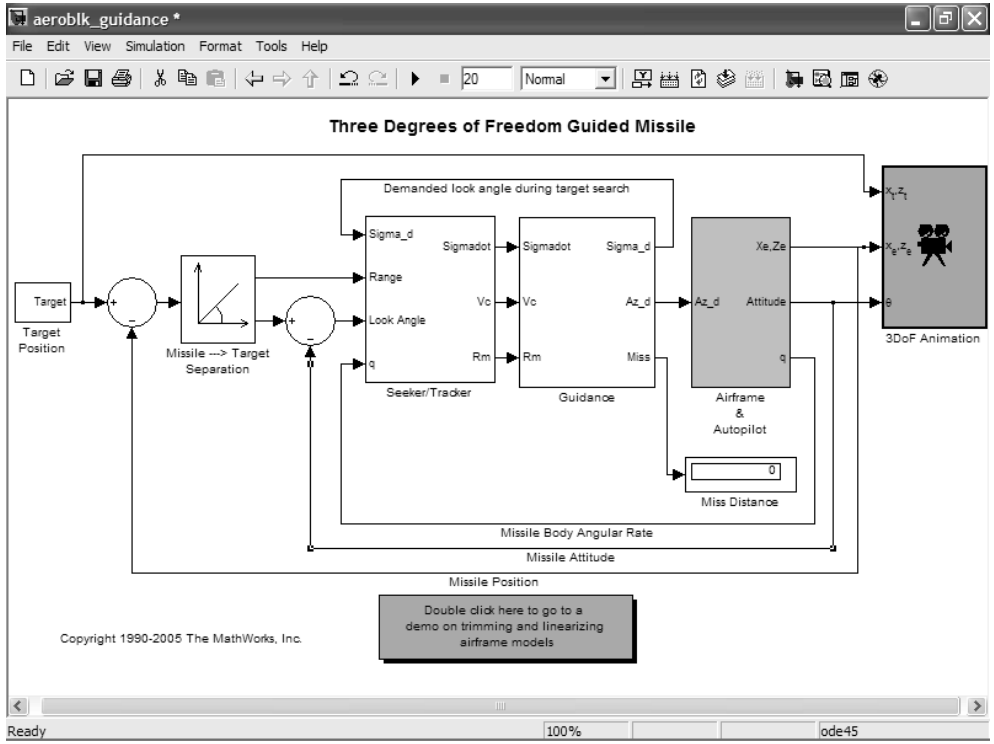


Рис. 12.92. Окно диаграммы примера aeroblk_guidance

12.9.6. Моделирование полета космического корабля – челнока

Закончить описание возможностей пакета Aerospace Blockset естественно на примере из нашей современности. Такой пример с именем aeroblk_HL20_VR есть в составе демонстрационных примеров, и диаграмма его представлена на рис. 12.96.

Средства визуализации посадки космического челнока представлены на рис. 12.97. Здесь даны кадр анимационной картины посадки челнока на посадочную полосу и осциллограммы, иллюстрирующие поведение аппарата.

Разумеется, пакет **Aerospace Blockset** имитирует лишь самое общее поведение таких сложных технических аппаратов, как самолеты, ракеты и космические аппараты. Это моделирование имеет учебный характер, хотя система MATLAB + Simulink используется для отдельных вычислений и моделирования узких задач в аэрокосмической промышленности и в научных организациях. Что касается серьезного моделирования таких аппаратов, то для этого нужно использовать самые мощные суперкомпьютеры и мощные программные комплексы специального назначения.

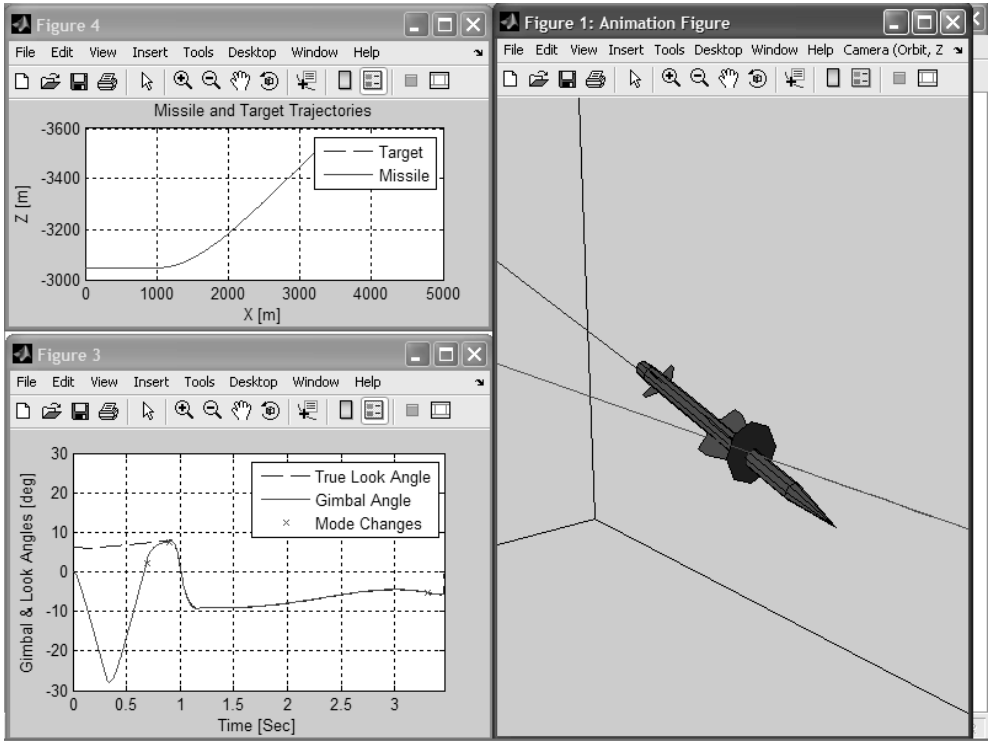


Рис. 12.93. Визуализация результатов моделирования примера aeroblk_guidance

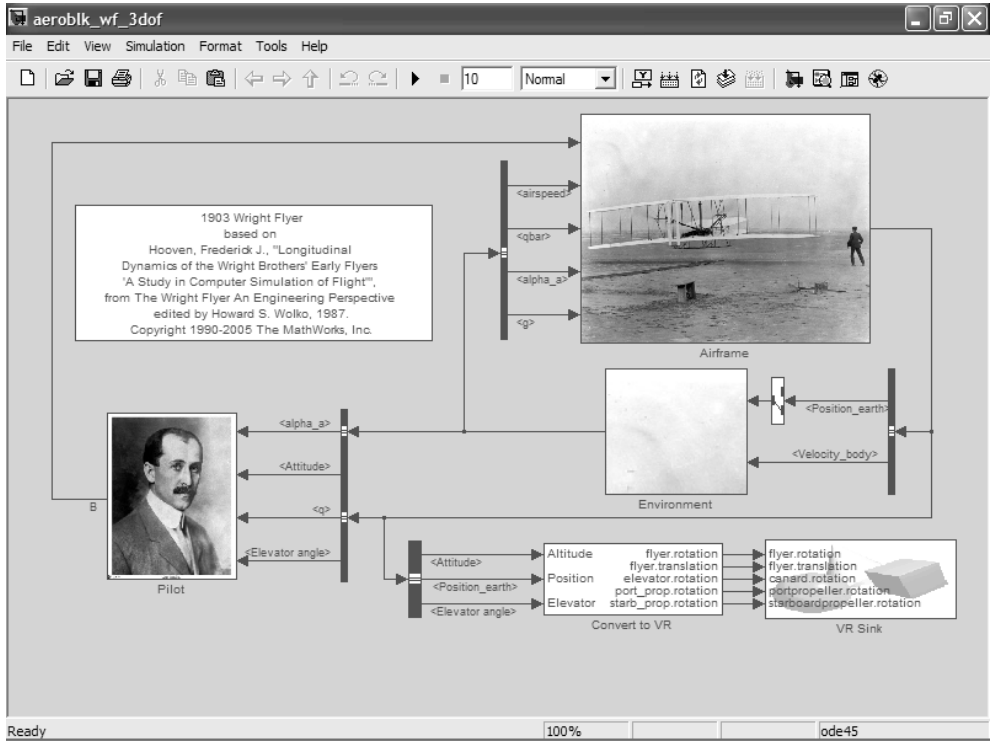


Рис. 12.94. Пример моделирования самолета – этажерки

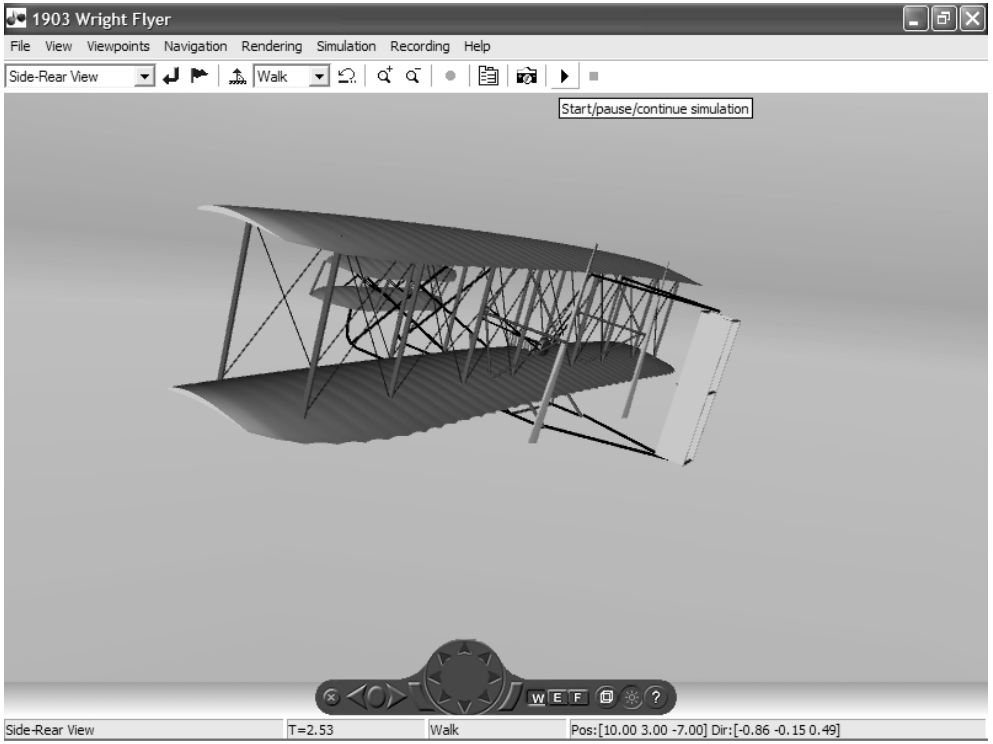


Рис. 12.95. Окно проигрывателя виртуальной реальности для примера моделирования полета самолета – этажерки

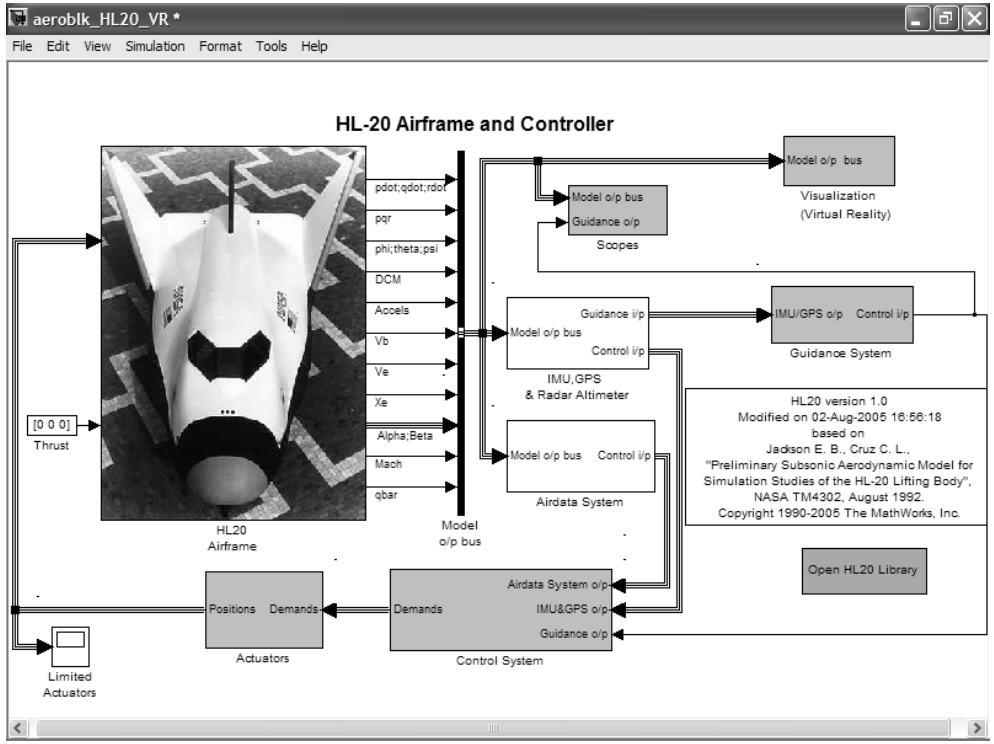


Рис. 12.96. Диаграмма моделирования полета космического челнока

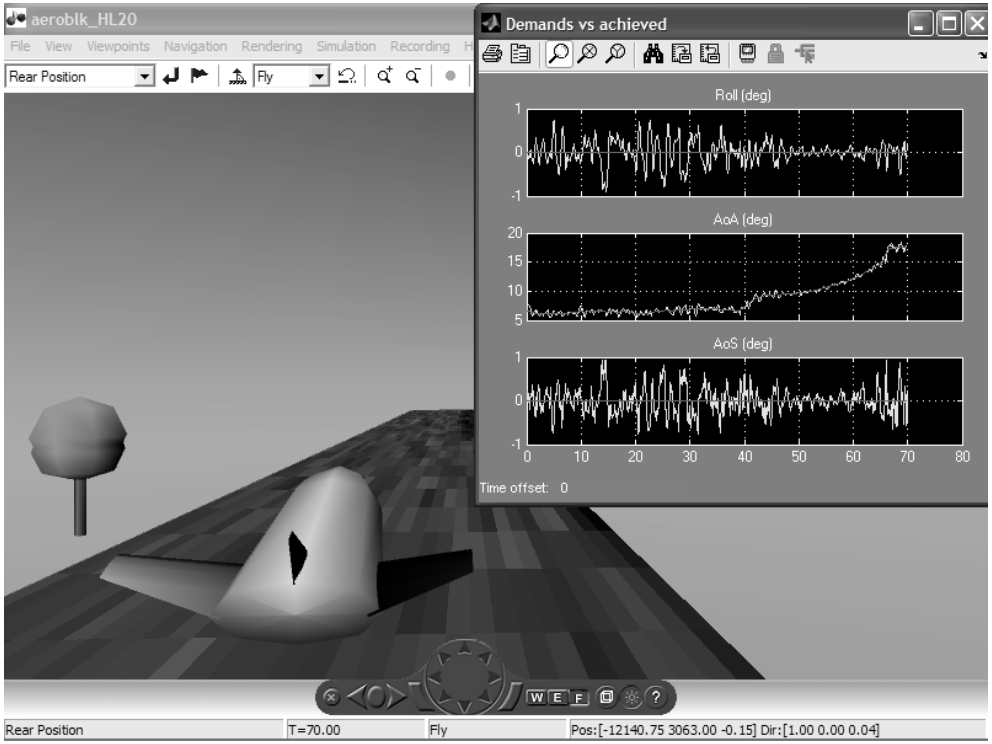


Рис. 12.97. Визуализация посадки космического челнока

Урок 13

Основы событийного моделирования

Событийное моделирование – совершенно новый тип моделирования, при котором структура модели может меняться в зависимости от событий, происходящих в процессе моделирования. Это кардинально отличает его от имитационного моделирования, реализованного в Simulink. Пакет расширения Simulink **Stateflow** обеспечивает (совместно с Simulink) проведение событийного моделирования. Этот урок является введением в пакет расширения **Stateflow**. На роль исчерпывающего описания этого сложного и мало у нас известного пакета данный урок не претендует.

13.1. Пакет Stateflow

13.1.1. Понятие о событийном моделировании

В этом уроке описывается пакет расширения Stateflow 5.*/6.*, придающий системе MATLAB + Simulink качественно новую возможность – осуществление событийного, или ситуационного, моделирования. Описание пакета носит обзорный характер.

Многие системы и устройства работают в условиях, когда возникновение тех или иных событий может существенным образом повлиять на их работу. Возьмем простой пример – источник энергии переменного тока (сигнала) подключен к нагрузке (индикатору). Моделирование покажет наличие на нагрузке соответствующего переменного напряжения. А теперь предположим, в духе нашего смутного времени, что поставщик электроэнергии неожиданно «вырубил рубильник» и отключил электроэнергию. Естественно, что работа нашей простой системы резко изменится – сигнал исчезнет, и напряжение на нагрузке станет равным нулю.

Моделирование систем и устройств, поведение которых зависит от внешних событий (ситуаций), называется *событийным*, или *ситуационным*, *моделированием*. В системе Simulink + MATLAB 6.* оно реализуется с помощью специального расширения Stateflow 5.*. В MATLAB 7.* он представлен версиями Stateflow 6.*. Однако в пределах начального знакомства с этим пакетом, составляющего цель данного урока, различие между версиями роли не имеет, и приведенный ниже материал может использоваться для знакомства с любой версией данного пакета.

13.1.2. Назначение пакета Stateflow

Пакет событийного моделирования **Stateflow** 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 человекомашинного 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.

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 для работы в реальном времени Real Time Workshop.

Эти компоненты обеспечивают интуитивно понятные и простые приемы работы с пакетом **Stateflow**.

13.1.3. Доступ к средствам Stateflow

Доступ к средствам **Stateflow** подобен уже описанному доступу к различным разделам библиотеки пакета Simulink. В окне библиотеки пакета **Stateflow** присутствует единственный блок **Chart**. Его можно перетащить мышью в окно модели Simulink для построения в дальнейшем SF-диаграммы. Для открытия окна с демонстрационными примерами надо выбрать пункт меню **File** ⇒ **Open** и найти файл **sf_exmples** в папке **toolbox\stateflow\sfdemos**. При загрузке этого файла откроется окно с демонстрационными примерами.

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. Первоначально активизация этого блока вызывает пустое окно редактора SF-диаграмм. Его можно использовать для создания SF-диаграмм и их отладки с целью получения от них нужных функций. Совокупность всех SF-блоков, имеющихся в составе модели, образует *Stateflow-машину*.

13.1.4. Понятие о SF-диаграмме

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 некоей абстрактной системы. Такая диаграмма с обозначенными на ней объектами представлена на рис. 13.1.

На рис. 13.1 показаны основные типы объектов SF-диаграмм. Ниже мы рассмотрим их более подробно.

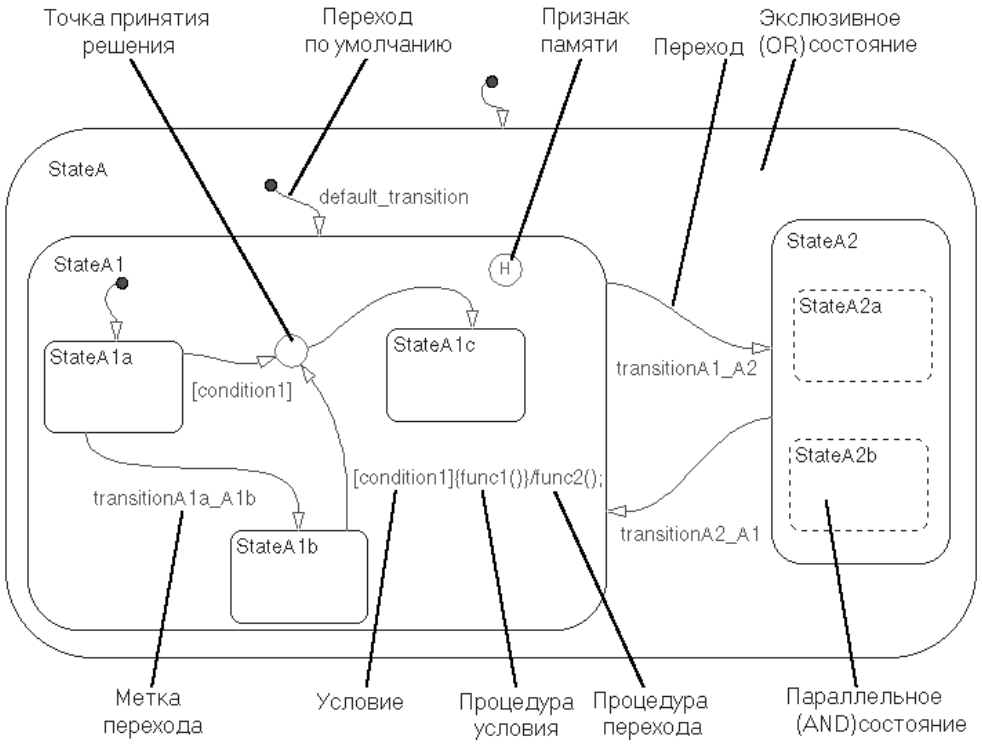


Рис. 13.1. Вид Stateflow-диаграммы

13.2. Основные объекты SF-диаграмм

13.2.1. Состояния и признаки памяти

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 состоянием*. 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».

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 в выполнении. Она изображается как красная окружность с буквой **H** внутри.

13.2.2. Переходы и признаки альтернативы

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.

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.

13.2.3. События, процедуры и данные

Событие (event) – еще одно 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 13.2 иллюстрирует построение двух состояний: нейтрального (**neutral**) и занятого (**engaged**). 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.

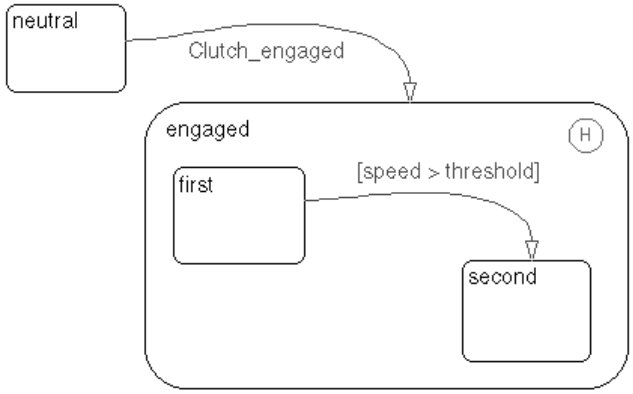


Рис. 13.2. Пример простой SF-диаграммы

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**, с помощью которого можно задать тип события и указать его свойства.

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. 13.2. Здесь можно выделить процедуру условия и процедуру перехода. Пример задания процедуры, связанной с состоянием объекта, будет дан чуть ниже.

При записи процедур используется семантика пакета **Stateflow**. Из-за ограниченного объема книги подробно семантика записи действий не рассматривается. В объемном фирменном описании пакета (896 страниц в формате PDF для версии 5.1.1) 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 все демонстрационные примеры, как включенные в этот урок, так и неотмеченные.

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 рабочей области 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 обозревателя **Stateflow**.

13.2.4. Описание объектов

При создании состояния в левом верхнем углу его графического изображения появляется знак вопроса. На его месте вводится описание состояния – в простейшем случае его имя. Общая структура состояния следующая:

```
name/
entry:
during:
exit:
on event_name:
```

Смысл этих определений дан ниже:

- `during:` или `du` – 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 относится к ключевому слову `broadcast` (в буквальном переводе радиопередача, а в нашем контексте означающее организацию связи между блоками без явного задания линии между ними).

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].
```

Пример описания состояния дан на рис. 13.3.

Этим примером мы воспользуемся позже при задании отмеченной в начале урока простой модели.

```
Power_on/
entry: ent_action();
during: dur_action();
exit: exit_action();
on Switch_off: on_action();
```

Рис. 13.3. Пример задания процедуры, связанной с состоянием объекта

13.3. Пример построения модели с SF-диаграммой

13.3.1. Работа с редактором SF-диаграмм

Итак, возьмем простой пример – источник переменного напряжения, подключенный к нагрузке – осциллографу. Реализуем эту модель в версии Stateflow. Зададим эту модель в окне Simulink и мышью перетащим в это окно блок **Chart**. Вначале этот блок при активизации мышью создает пустую SF-диаграмму – заготовку для создания действующей диаграммы. Если активизировать мышью пустой блок **Chart**, то появится окно редактора SF-диаграмм. Перетаскивая мышью пиктограммы панели инструментов, находящейся в правой части окна редактора, можно поместить в SF-диаграмму нужные компоненты – см. рис. 13.4 – с построенной простой SF-диаграммой для нашей модели.

Редактор SF-диаграмм имеет традиционный для окон системы MATLAB вид. Основные средства построения диаграмм сосредоточены в меню **Add (Добавить)**.

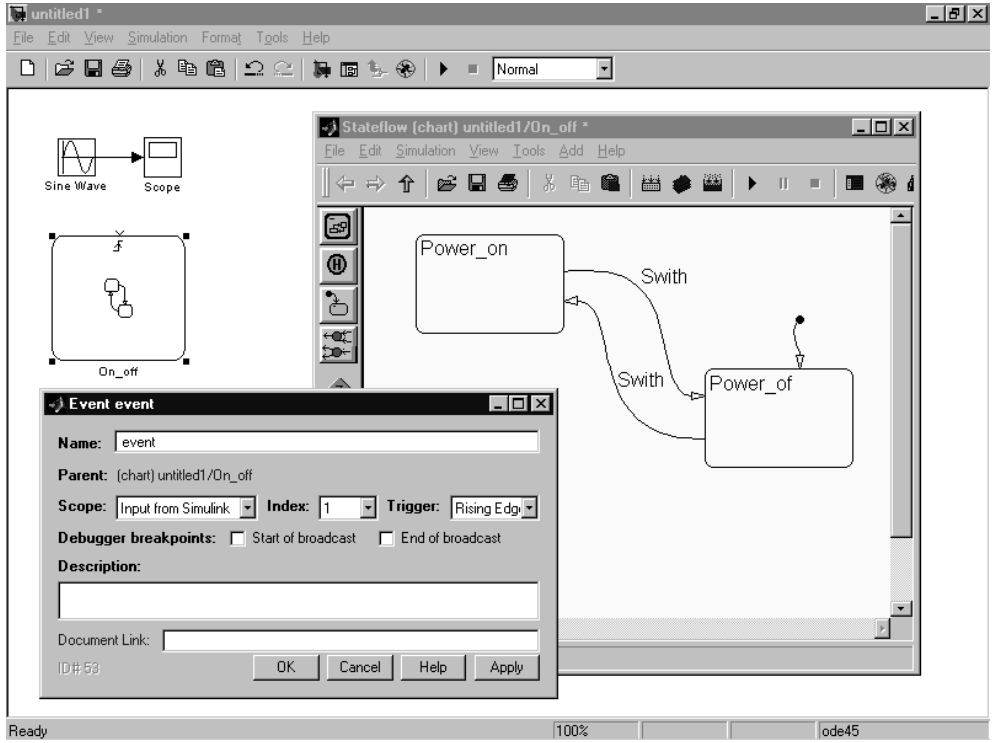


Рис. 13.4. Создание простой модели и ее SF-диаграммы

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. 13.5. Если установить курсор мыши на той или иной кнопке, то ее название появляется в строке состояния окна редактора SF-диаграмм.

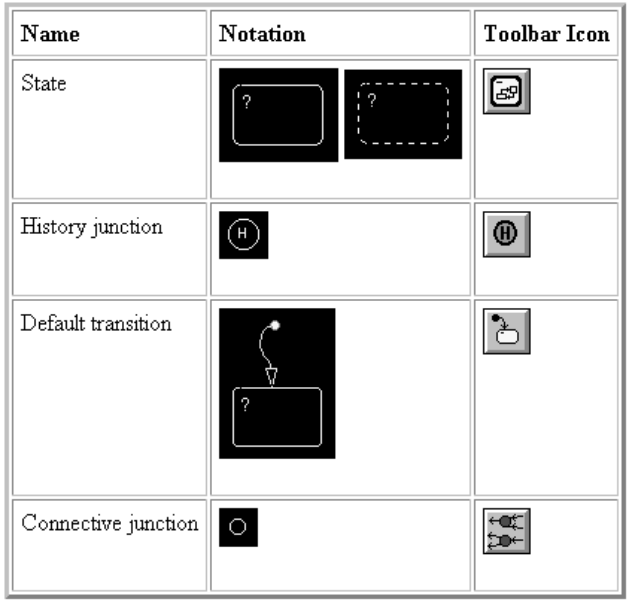


*Рис. 13.5. Средства построения SF-диаграммы пакета **Stateflow***

Блоки можно переносить мышью в окно SF-диаграммы, а затем соединять их друг с другом с помощью мыши при нажатой левой кнопке. 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.

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 к 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**.

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.

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**. 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.

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. 13.4). Пока можно согласиться с параметрами, заданными по умолчанию. Для этого достаточно нажать клавишу **OK**.

Если это сделано, то над блоком SF-диаграммы появится характерный триггерный вход, с которым можно соединить выход генератора сигнала. Это будет означать, что мы связали работу генератора сигнала с SF-диаграммой. Теперь наша система «источник энергии – получатель энергии» попадает в зависимость от наличия или отсутствия энергии, то есть система становится событийно управляемой.

13.3.2. Установка параметров SF-диаграммы с помощью обозревателя

Еще один способ установки параметров предоставляет обозреватель Stateflow Explorer (рис. 13.6). Для его запуска используется команда меню **Tools** ⇒ **Explorer...**.

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** необходимо 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.

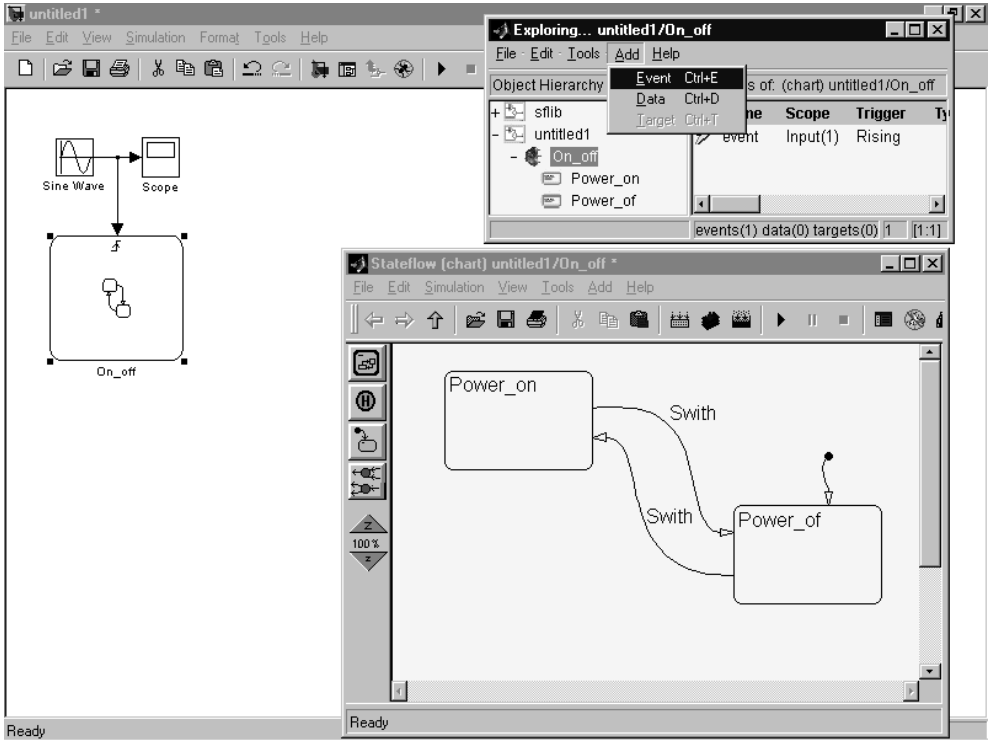


Рис. 13.6. Установка параметров SF-диаграммы в окне обозревателя

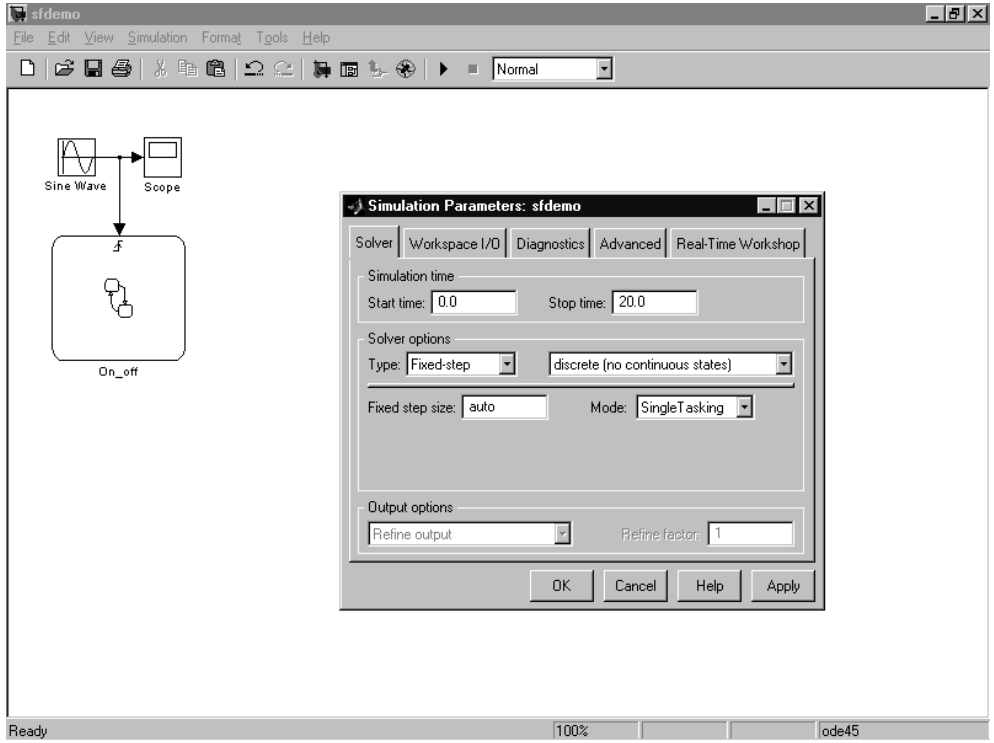


Рис. 13.7. Установка параметров запуска

13.3.3. 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.

13.4. Запуск, отладка и форматирование SF-диаграмм

13.4.1. 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. 13.7.

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).

13.4.2. 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.

Однако пока вместо всего этого мы увидим ряд окон с указанием на обнаруженную в диаграмме ошибку – неверную запись команды **Switch**. Эту ситуацию характеризует рис. 13.8.

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

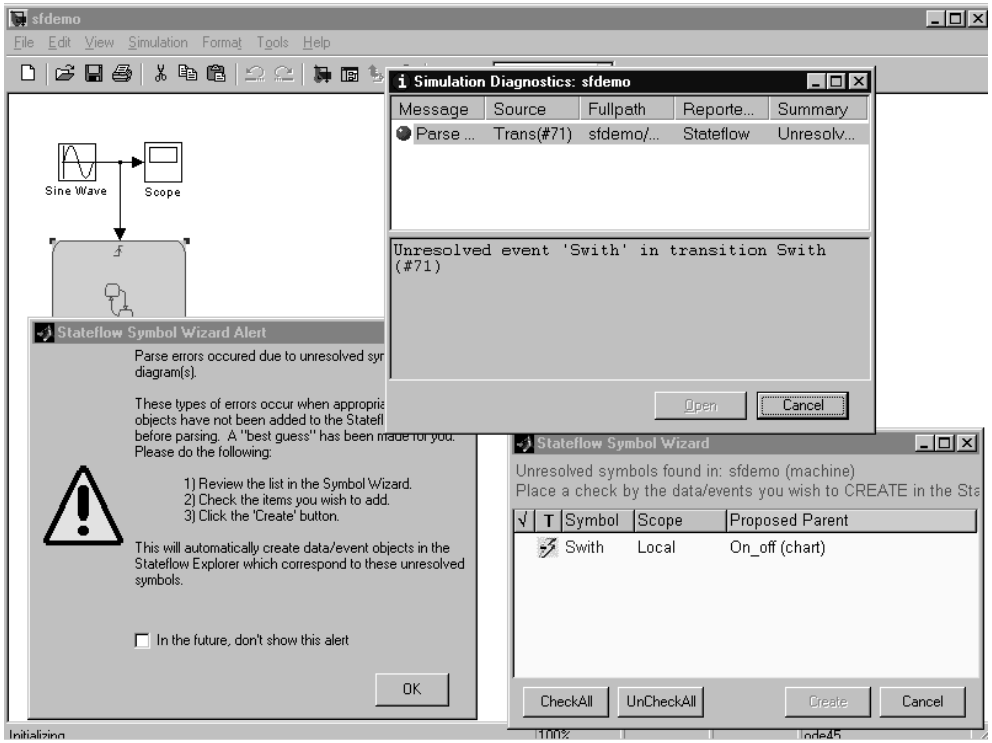


Рис. 13.8. Сообщения об ошибке после запуска модели

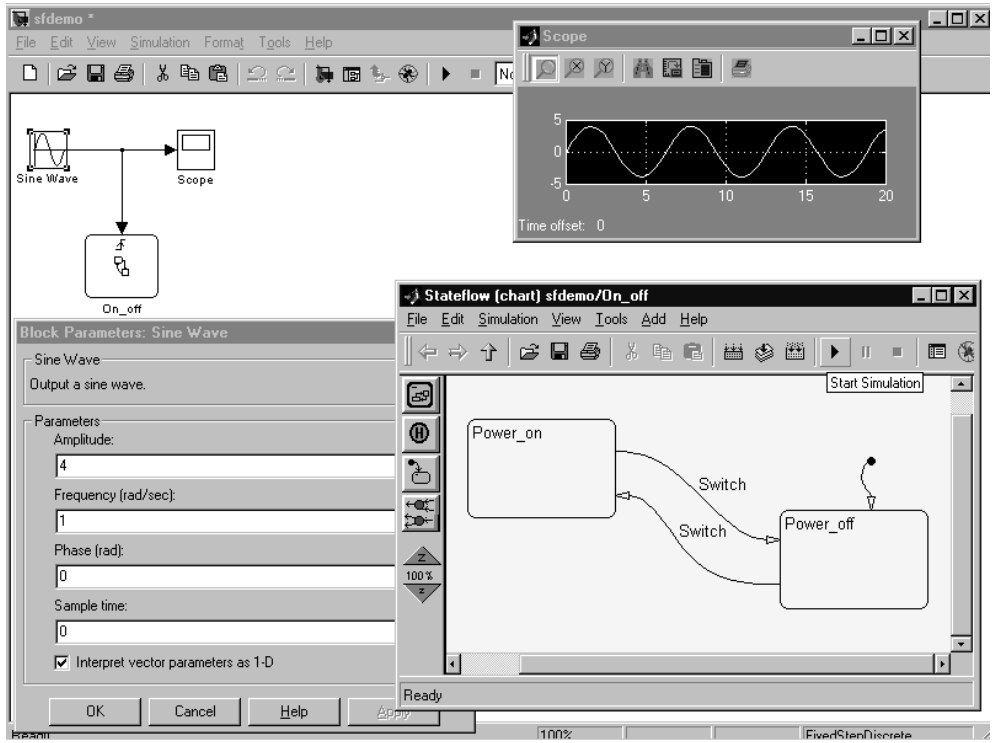


Рис. 13.9. Окончательный вид модели выключателя с SF-диаграммой

Power_off (отсутствие последней **f**). Проверьте и при необходимости скорректируйте установки параметров. Модель будет иметь вид, представленный на рис. 13.9.

Теперь наша модель должна запускаться как из окна модели Simulink, так и из окна редактора SF-диаграмм.

13.4.3. 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. 13.10) появляется при выполнении в окне SF-диаграммы команды меню **Tools** ⇒ **Debug...**.

Рисунок 13.10 иллюстрирует начало моделирования при нажатии кнопки **Start** отладчика (она после этого меняет название на **Continue**). Нетрудно заметить, что сначала активизируются 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 13.11 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. 13.12 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.

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

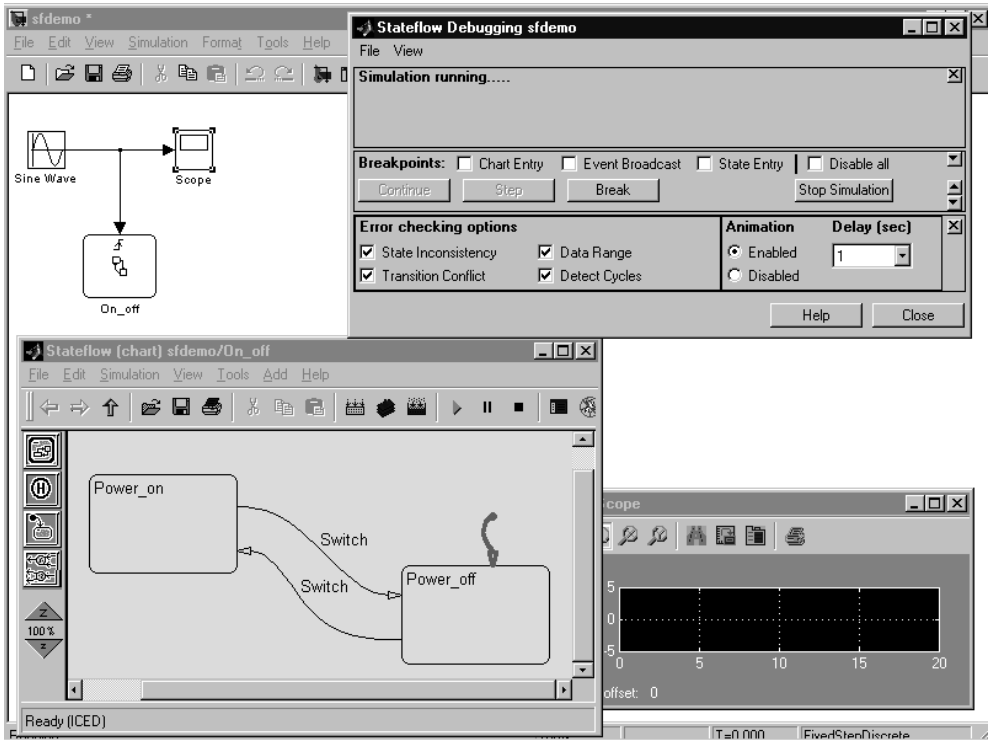


Рис. 13.10. Начало моделирования

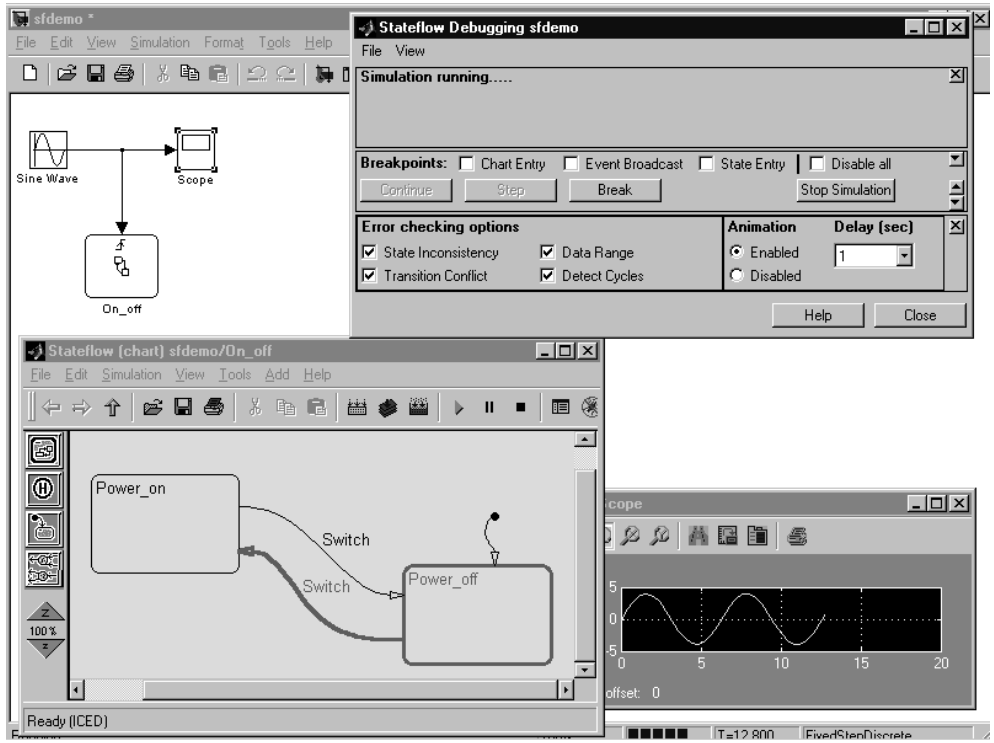


Рис. 13.11. Промежуточный кадр SF-диаграммы

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.

13.4.4. 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. 13.13 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.

Для выделенного в окне анализатора сообщения можно открыть окно построителя целевого кода – **Simulation Target Builder** (рис. 13.13). Краткое описание вопросов генерации кодов будет дано в конце этого урока.

Кнопка **Target Options** открывает окно опций построителя целевого кода. На рис. 13.13 оно расположено в верхнем левом углу с открытым списком возможных опций. Другая кнопка – **Coder Options** – открывает окно опций специального

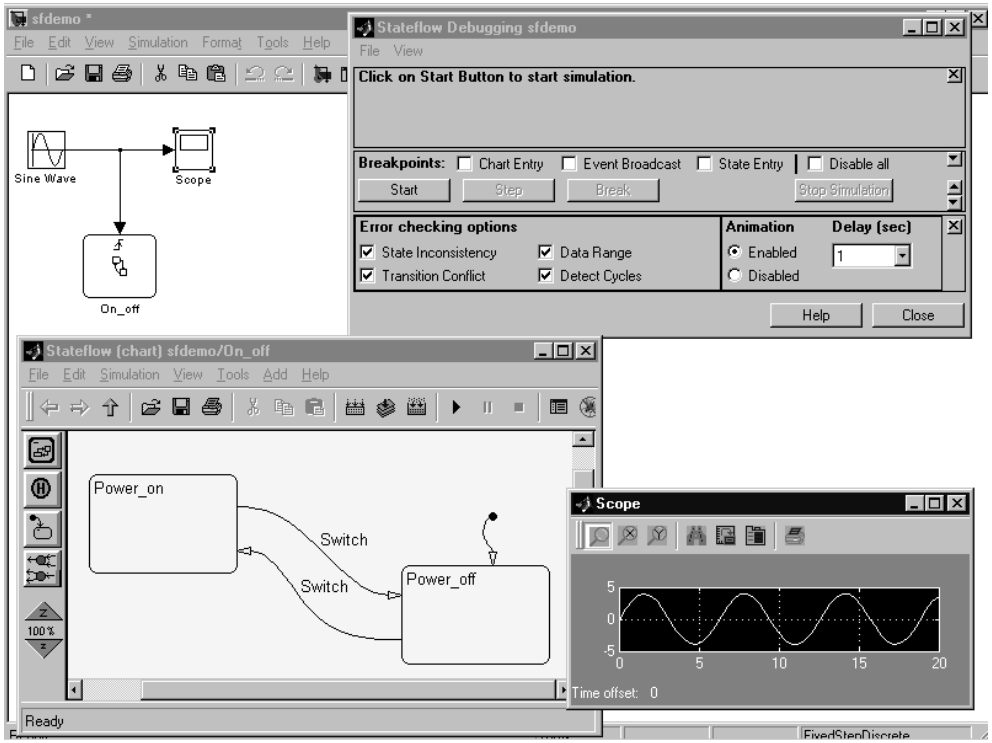


Рис. 13.12. Модель с SF-диаграммой в конце моделирования

компонента Stateflow – генератора программного кода Stateflow Coder. 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 представления сообщений о создавамой после нажатия кнопки **Build** DF-диаграмме.

Таким образом, рис. 13.13 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.

13.4.5. 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** ⇒

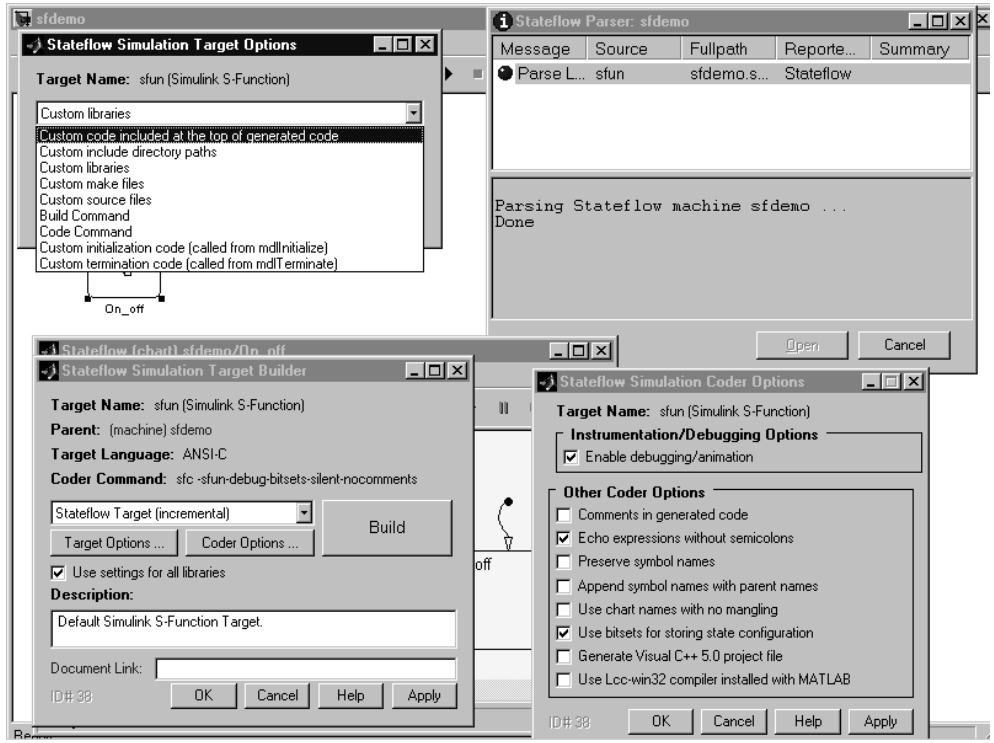


Рис. 13.13. Средства отладки SF-диаграмм

Find... редактора SF-диаграмм. Эта команда открывает окно поиска объектов, в котором можно задать имя разыскиваемого объекта. Нажав кнопку **Find**, можно получить список всех объектов, в которых обнаружено заданное имя.

13.4.6. Выбор стиля SF-диаграмм

Размер объектов на SF-диаграммах можно менять с помощью переключателя под кнопками вывода графических объектов в панели инструментов. 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. Примеры задания стилей объектов имеются в окне выбора стиля.

13.4.7. 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.

13.5. Особенности версий пакета расширения Stateflow

13.5.1. Новый редактор SF-диаграмм в Stateflow 5.*

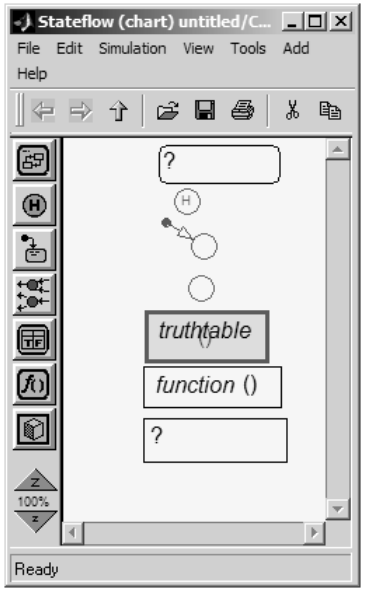


Рис. 13.14. Окно редактора SF-диаграмм пакета Stateflow 5.1.1

Как уже отмечалось, пакет **Stateflow** быстро развивается. Так, уже в версии Stateflow 5.1.1, применяемой в системе MATLAB 6.5 SP + Simulink 5, возможности пакета были заметно расширены. К уже описанным кнопкам редактора SF-диаграмм были добавлены еще три кнопки (рис. 13.14):

- установка таблицы обмена `Truche Table`;
- установка функции `Function`;
- установка ящика `Box`.

Добавление этих средств заметно расширяет возможности построения SF-диаграмм.

13.5.2. Несколько простых примеров применения Stateflow 5.*

Продолжим знакомство с пакетом Stateflow 5.1.1. Весьма наглядный пример применения конструкции If представлен на рис. 13.15. Здесь SF-диаграмма задает блок условных выражений, исполнение которых зависит от уровня входного сигнала, заданного источником линейно-нарастающего сигнала Ramp. Если уровень сигнала condition менее или равен 20, то задается выход сигнала, возведенного в квадрат. На интервале от 20 до 50 задается вывод синусоидального сигнала 50*sin(condition). А при condition>50 выводится сам сигнал. Осциллограммы на входе и выходе SF-диаграммы наглядно показывают ее работу.

В демонстрационных примерах пакета Stateflow 5.1.1 можно найти подобные примеры на реализацию циклов for, do и while. На рис. 13.16 показан пример загрузки синусоидального сигнала в блок SF-диаграммы из рабочего пространства MATLAB и выдачи его после задержки на 20 с.

Операция векторизации с помощью SF-диаграммы представлена на рис. 13.17. Блок SF-диаграммы здесь обеспечивает прием двух четырехэлементных векторов данных data1 и data2 и выдает вектор той же длины output=data1*data2+1. Выходной вектор контролируется цифровым дисплеем.

Рисунок 13.18 демонстрирует модель простой системы двухстороннего контроля температуры бойлера. На рисунке приведена SF-диаграмма двухстороннего

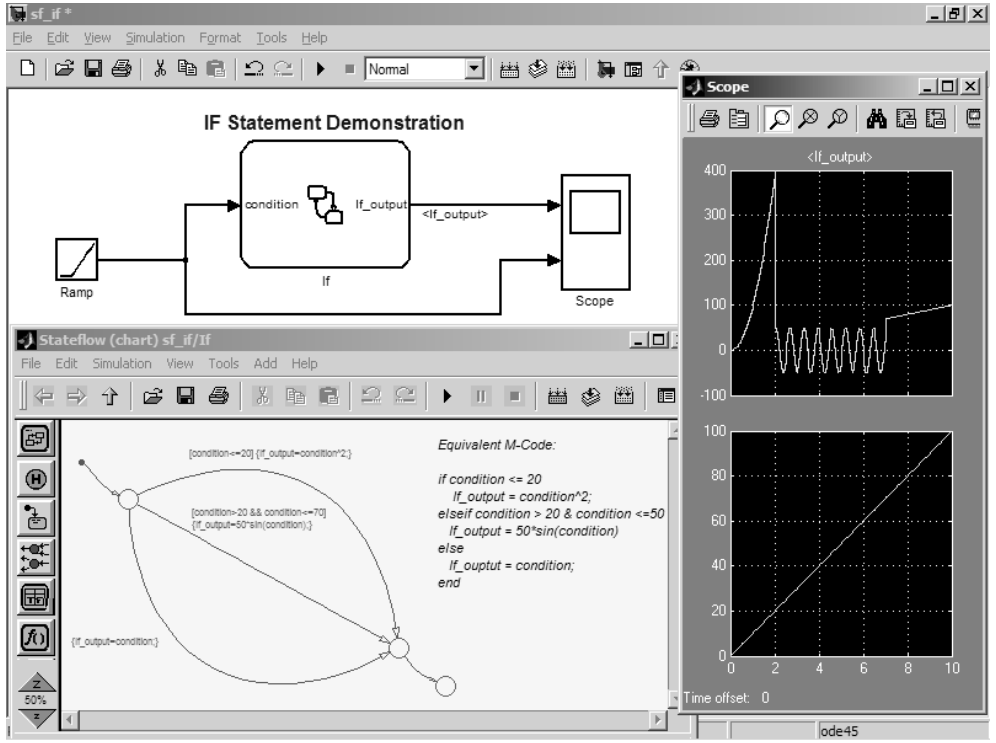


Рис. 13.15. Демонстрация конструкции If с помощью SF-диаграммы

контроллера температуры. Обратите внимание на применение новых средств – функций. Осциллограммы наглядно иллюстрируют релаксационный характер установления температуры.

13.5.3. Пакет расширения Stateflow 6.*

В новых реализациях системы MATLAB 7.* + Simulink 6.* используются обновленные версии пакета расширения Stateflow 6.*. Так, в последней реализации MATLAB 7.1 + Simulink 6.3 SP 3 используется версия Stateflow 6.3. С ней можно ознакомиться по справке, окно которой представлено на рис. 13.19.

На рис. 13.19 приведена страница справки, в которой перечислены новинки последней версии пакета. Это разделение глобальных данных между Simulink и Stateflow, поддержка средств работы с Fixed Point (числами с фиксированной точкой, лежащими в основе быстрой аппаратной реализации вычислений), применение специальных таблиц Truch Tables и др. Применительно к задачам этого урока эти новые средства значения не имеют.

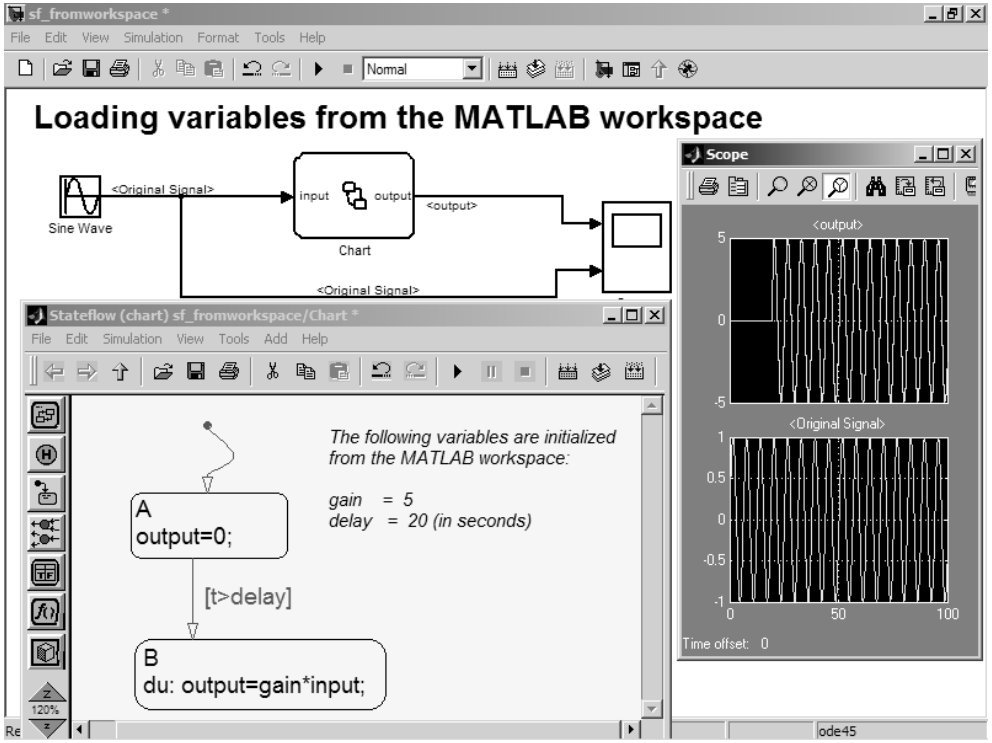


Рис. 13.16. Демонстрация загрузки синусоидального сигнала из рабочего пространства MATLAB и выдачи его с задержкой

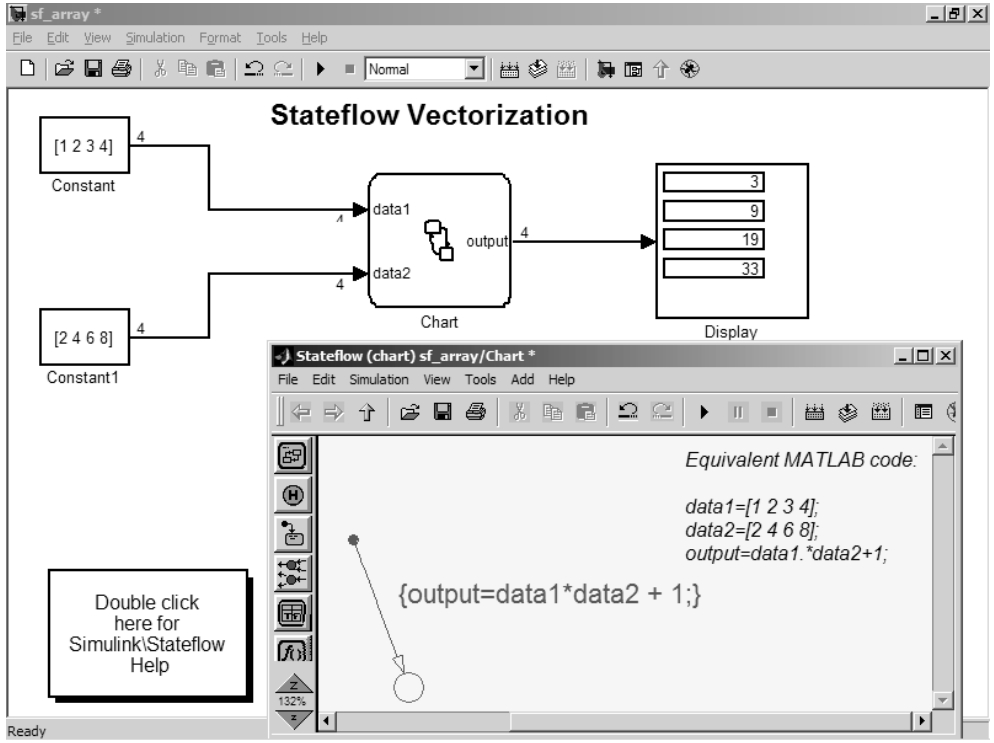


Рис. 13.17. Пример векторизации с помощью SF-диаграммы

13.6. Примеры применения пакета Stateflow 6.3

13.6.1. Работа с демонстрационными примерами

Пакет Stateflow имеет множество интересных и поучительных примеров, несколько их которых мы рассмотрели выше. Продолжим это знакомство на уровне более сложных примеров. Доступ к ним осуществляется из вкладки **Demos** справки – рис. 13.20.

При ознакомлении с демонстрационными примерами пакета **Stateflow** с учетом их сложности (взаимодействие ряда систем) целесообразно придерживаться следующих рекомендаций:

- загрузите пример в окно пакета **Simulink**;
- ознакомьтесь с моделью примера и выясните наличие в ней субмоделей;
- просмотрите блок-схемы субмоделей и разберитесь с их работой;
- сделайте пробный запуск примера;
- если моделирование идет слишком долго, остановите его и проверьте установку параметров моделирования (в частности, конечного времени) и измените параметры;
- снова запустите пример на моделирование;
- по завершении моделирования выведите осциллограммы и иные иллюстрации работы модели, предусмотренные в ней;
- постарайтесь осмыслить полученные результаты;
- выведите SF-диаграмму и запустите ее на исполнение (лучше делать это с помощью отладчика SF-диаграмм);
- проанализируйте SF-диаграмму, отметьте новые для вас детали и разберитесь с ними;

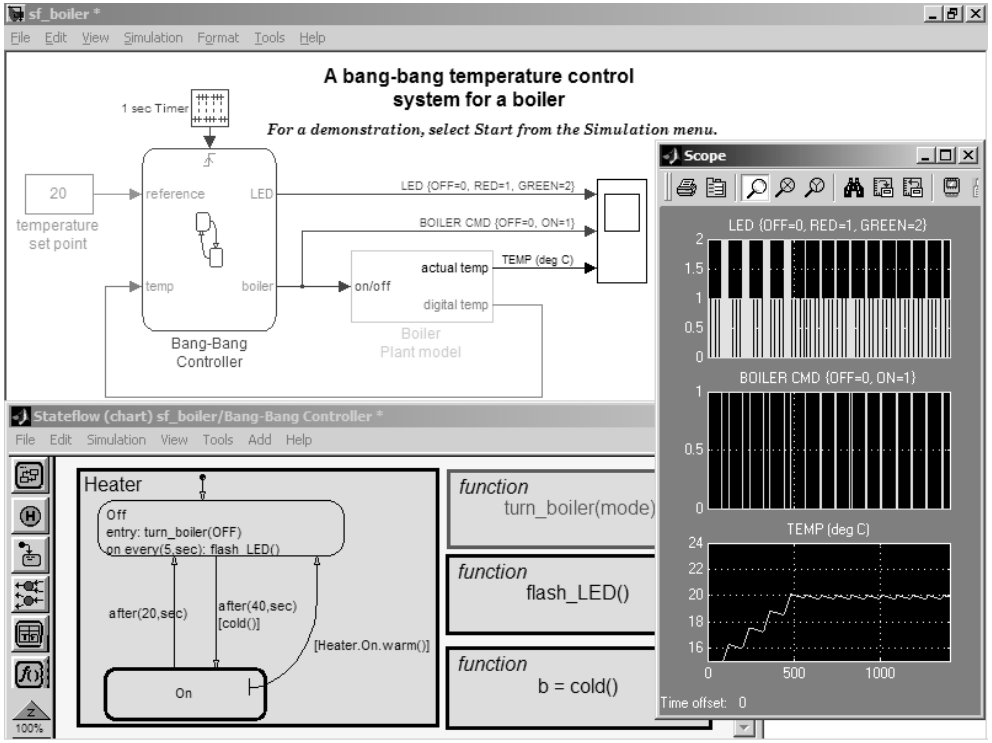


Рис. 13.18. Пример контроля температуры бойлера с помощью SF-контроллера

- 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 (а то и дней) 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 достаточно быстро и надежно.

Ниже представлено несколько весьма поучительных (как для знакомства с техникой моделирования в пакете Simulink, так для применения пакета Stateflow) избранных примеров. Мы рассмотрим их в порядке повышения сложности моделей.

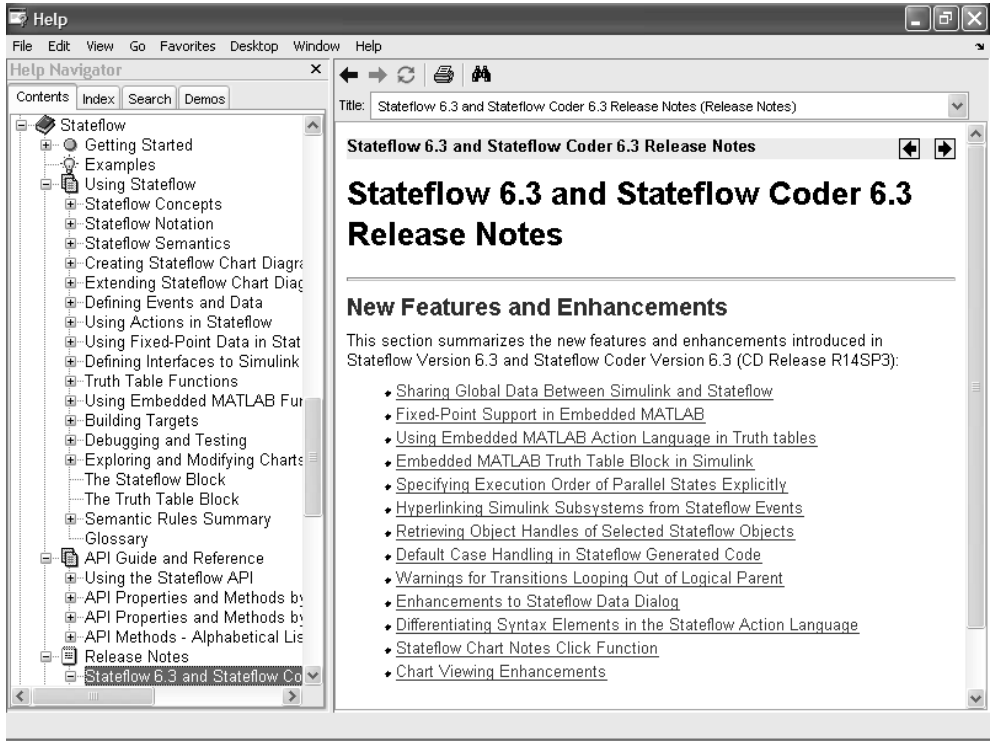


Рис. 13.19. Страница справки по пакету расширения Stateflow 6.3

13.6.2. Пример реализации рекурсивной функции вычисления факториала

В разделе **Graphical Functions** раздела **Demos** справки имеются три простых примера на реализацию графических функций. Рассмотрим один из них – реализацию рекурсивной функции факториала. Simulink-модель и Stateflow-диаграмма этого примера показаны на рис. 13.21. Результат вычислений отображается на экране виртуального дисплея.

Этот пример реализует рекурсивное вычисление факториала по известной формуле n!=n*(n-1)!. Она может дать огромный выигрыш в скорости вычислений последовательных значений (таблицы) факториалов. Этот пример настолько прост, что пользователю стоит повторить его самостоятельно и внимательно разобраться в динамике Stateflow-диаграммы. Вы можете также составить свой пример на другие рекурсивные вычисления, например чисел Фибоначчи.

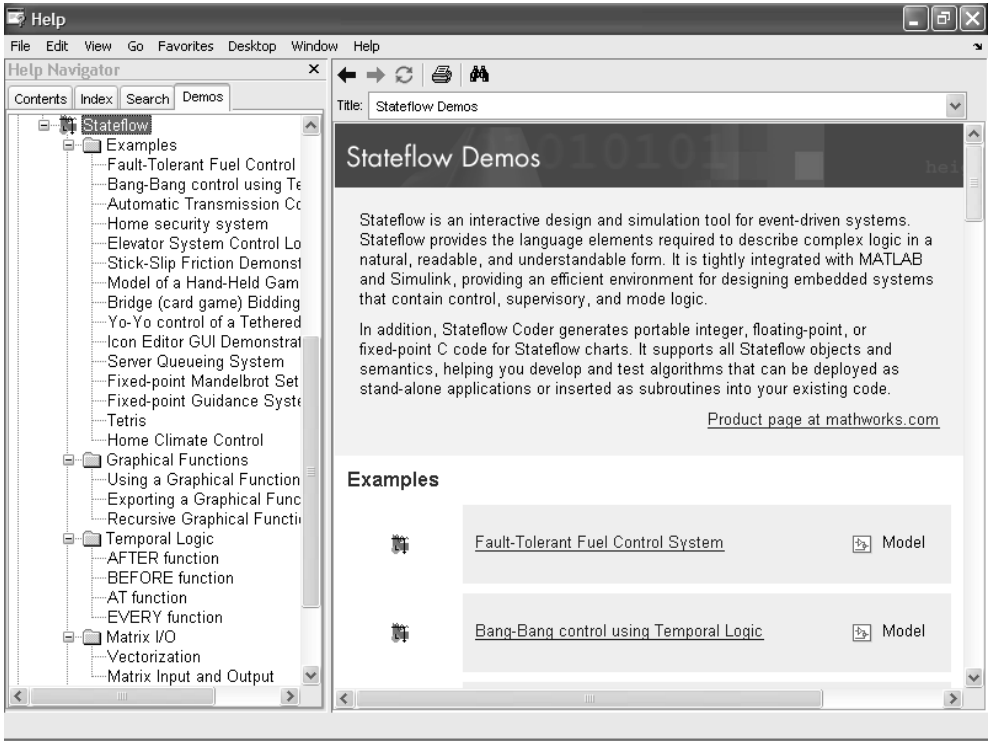


Рис. 13.20. Доступ к демонстрационным примерам Stateflow 6.3

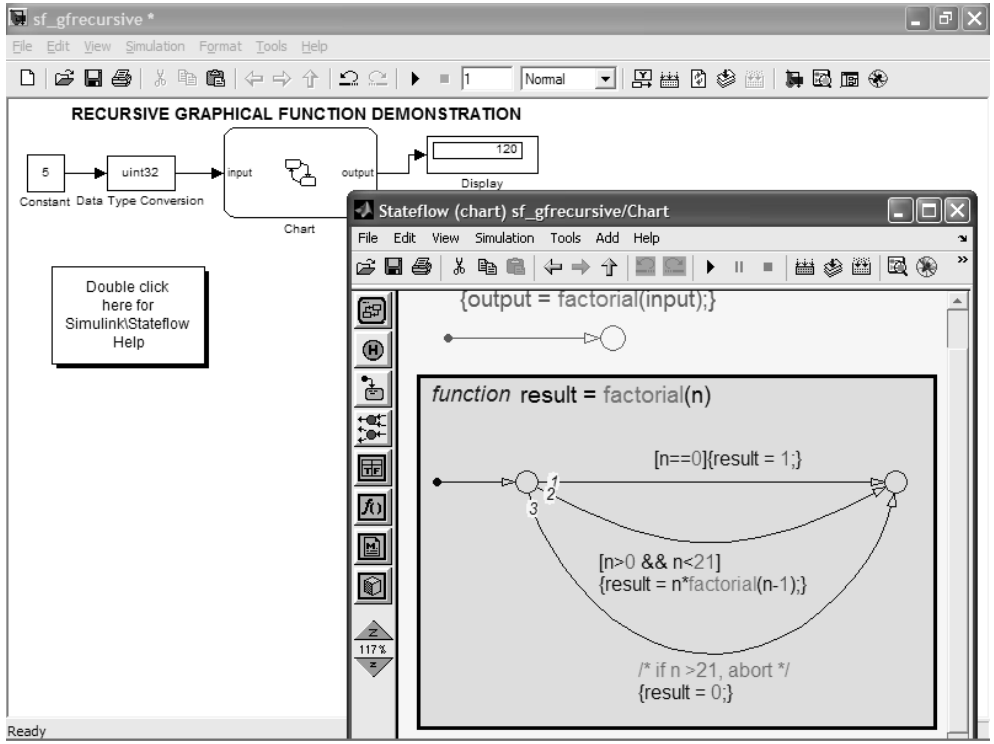


Рис. 13.21. Пример реализации вычисления факториала по рекурсивной формуле

13.6.3. Пример векторизации

Еще один наглядный пример на выполнение операций с векторами представлен на рис. 13.22. Simulink-диаграмма тут не нуждается в особом описании – она предельно проста. Реализуемая ею MATLAB-функция представлена сверху Stateflow-диаграммы. Необходимая формула векторизации реализуется Stateflow-диаграммой.

Здесь в качестве многопараметрического измерителя использован виртуальный осциллограф. Он измеряет постоянные уровни «сигналов», которые отображаются горизонтальными линиями. Аналогичный пример на операции с матрицами вы можете просмотреть самостоятельно.

13.6.4. Пример организации цикла

Еще один простой и очень наглядный пример применения Stateflow-диаграммы представлен на рис. 13.23. Здесь эта диаграмма задает вычисление цикла с управляющей целочисленной переменной i, меняющейся от 1 до 10 с шагом 1. Этот пример не нуждается в особых комментариях. Познакомьтесь с другими примерами такого рода в папке **Loop Statement** вкладки **Demos** справки по пакету расширения **Stateflow**.

13.6.5. Пример работы с Fixed Point средствами

В ряде случаев полезно применение числовых данных с фиксированной точкой – Fixed Point (FP) – см. пример на рис. 13.24. Операции с такими данными MATLAB выполняет предельно быстро и точно, используя при этом аппаратные средства на уровне микропроцессоров ПК. В ряде пакетов расширения предусмотрен переход на FP-модели и данные. Например, такие модели используются при проектировании активных и цифровых фильтров, которые нуждаются в последующей аппаратной реализации. При этом использование FP-данных способно заметно улучшить характеристики моделируемых устройств и проще воплотить их «в металл».

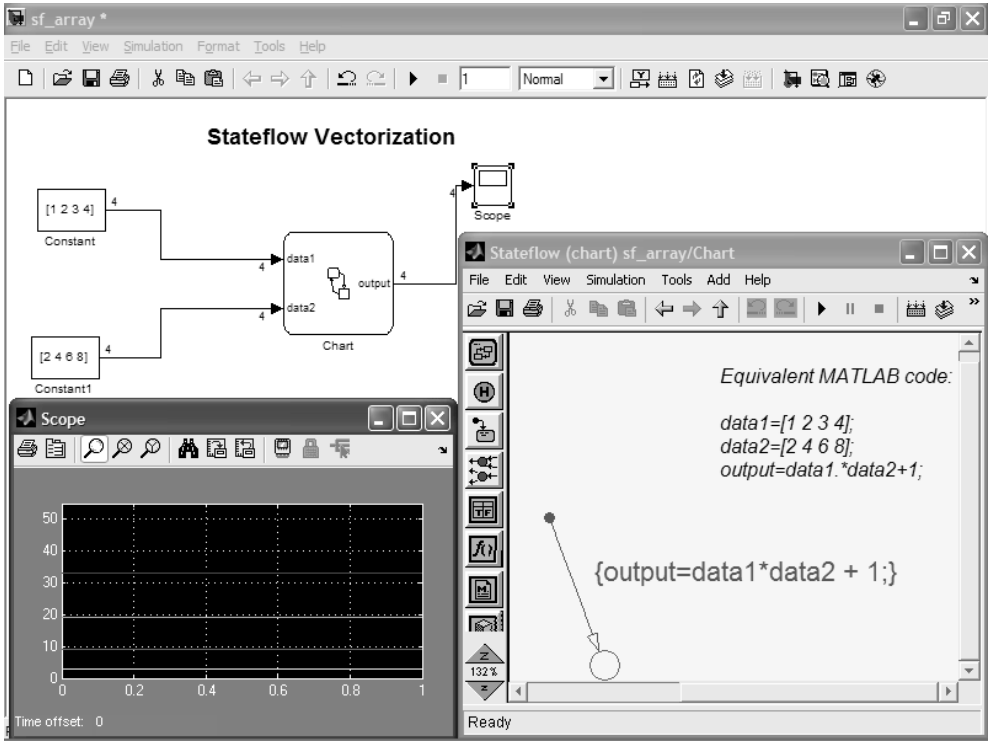


Рис. 13.22. Пример выполнения операций над векторами – векторизации

В этом примере (см. осциллограммы) переход к FP-данным привел к довольно выраженному квантованию выходного сигнала Stateflow-диаграммы. Сигнал при этом заметно искажается. Для сведения искажений к разумному пределу надо увеличивать число ступеней квантования.

13.6.6. Пример работы с рабочим пространством MATLAB

Как известно, все переменные (скалярные, векторные, матричные и строковые) располагаются в рабочем пространстве системы MATLAB. Актуален доступ к ним из Stateflow-диаграмм. На рис. 13.25 представлен простой пример, иллюстрирующий решение этой задачи.

В этом примере Stateflow-диаграмма выполняет функции реализации идеального усилителя, имеющего коэффициент передачи gain=5 и время задержки delay=20 с. При этом переменные gain и delay хранятся в рабочем пространстве MATLAB и вызываются из него в ходе исполнения Stateflow-диаграммы. На вход

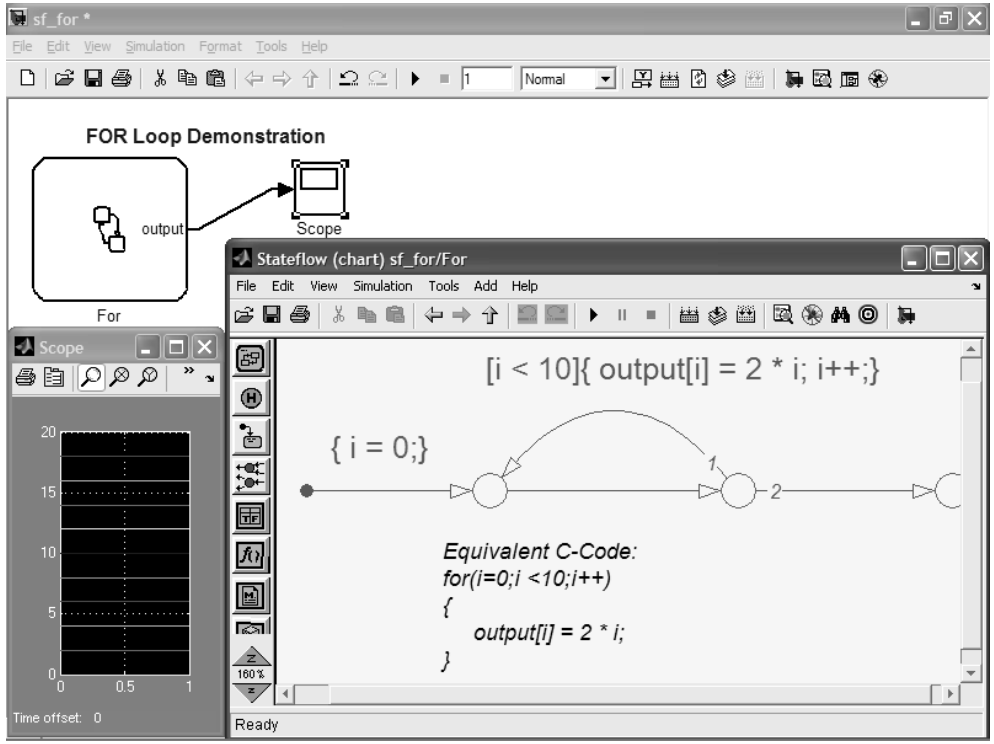


Рис. 13.23. Пример реализации цикла типа for

«усилителя» подан синусоидальный сигнал, который контролируется в нижнем окне виртуального осциллографа. В верхнем окне показан сигнал на выходе Stateflow-блока. Он задержан на время delay и увеличен по амплитуде в gain раз.

Еще один пример работы с рабочим пространством дан на рис. 13.26. Там же даны диаграммы субмоделей и результаты работы.

13.6.7. Построение фрактала Мандельброта

Фрактал Мандельброта – одна из самых загадочных и красивых математических фигур. Рисунок 13.27 демонстрирует построение этой фигуры с помощью SF-диаграммы для двух вариантов – с применением арифметики с двойной точностью и арифметики чисел с фиксированной разрядностью.

Даже умеренно сложные SF-диаграммы плохо видны на страницах книг, поскольку содержат множество мелких деталей. Их желательно наблюдать на экране дисплея ПК с высоким разрешением. Это позволяет наблюдать также анимационные цветные диаграммы, которые характерны для ряда примеров.

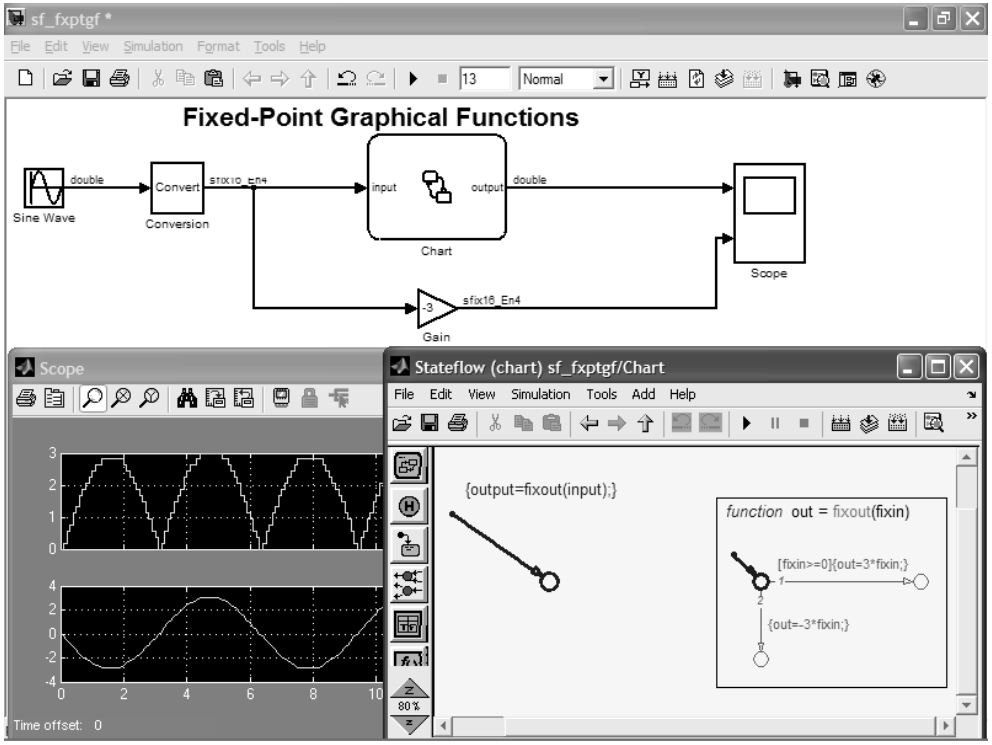


Рис. 13.24. Пример применения средств работы с FP-данными

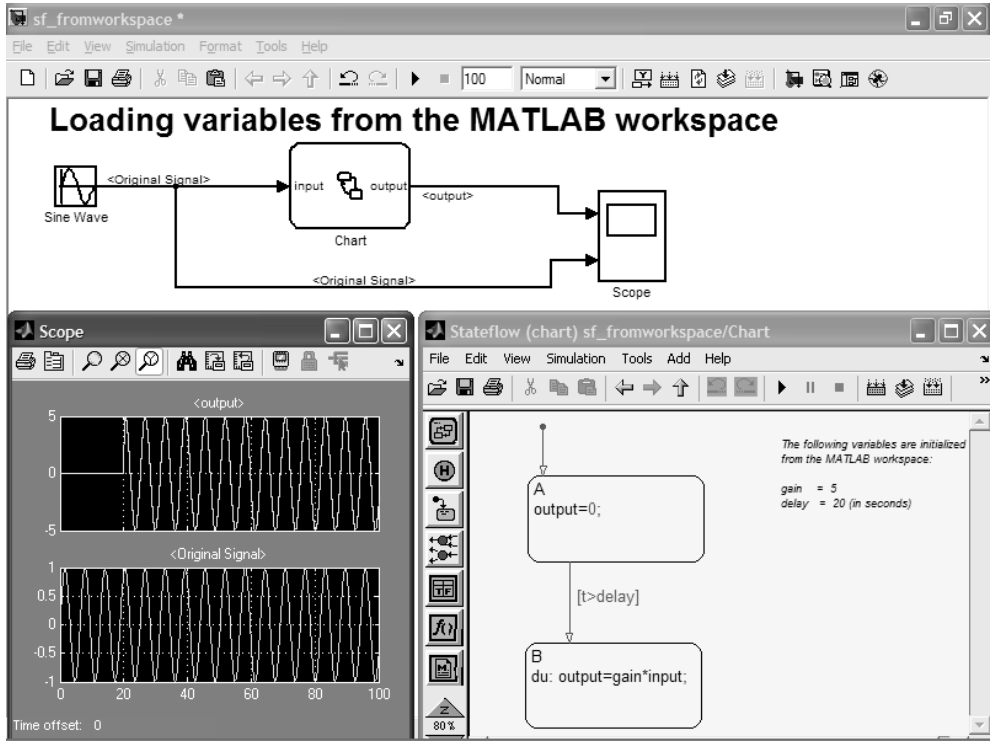


Рис. 13.25. Пример работы с рабочим пространством системы MATLAB

13.6.8. 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 направление, то трение порождает гистерезис зависимости положения бруска от действующей на него силы.

Simulink-модель этой простой физической системы показана на рис. 13.28. Там же представлены результаты моделирования – осциллограммы временной зависимости силы и перемещения бруска и фазовый портрет его движения, явно показывающий отмеченный выше гистерезис. На рис. 13.28 представлены также осциллограммы «сигналов», иллюстрирующие характер движения бруска.

SF-диаграмма этой несложной модели также предельна проста, и это позволяет легко разобраться с ее особенностями. Она и подсистема Mechanica Motion показаны на рис. 13.29. Показано также окно редактирования параметров, которое появляется при активизации блока edit parameters.

13.6.9. Моделирование системы трансмиссии автомобиля

Приведем пример моделирования технического устройства – рис. 13.30. На этот раз моделируется электрогидравлический сервомеханизм. Такие механизмы широко применяются в промышленности и на транспортных средствах.

Вы можете самостоятельно ознакомиться с деталями работы этой модели. Ее SF-диаграмма представлена на рис. 13.31.

Для детального знакомства со сложными моделями и SF-диаграммами очень полезен обозреватель моделей – Model Explorer. На рис. 13.32 показан пример применения обозревателя SF-диаграмм для модели, представленной на рис. 13.30

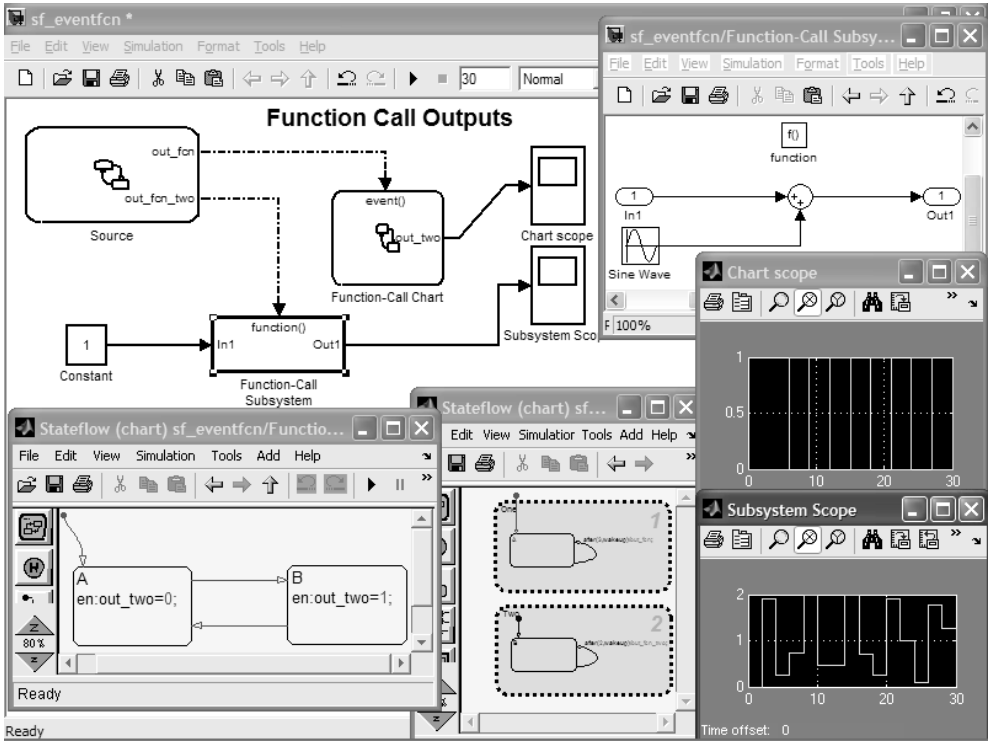


Рис. 13.26. Еще один пример работы с рабочим пространством

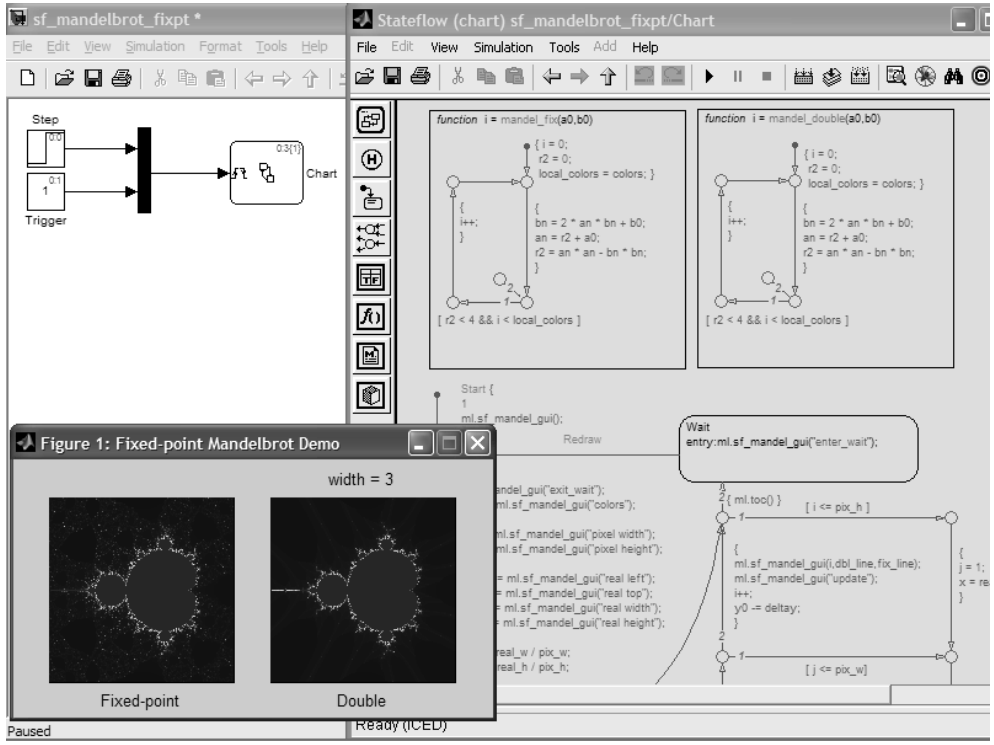


Рис. 13.27. Модель построения фрактала Мандельброта

и 13.31. Чем сложнее модель, тем целесообразнее получение данных о ней с помощью обозревателя.

13.6.10. 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 13.33 показывает модель после ее запуска и вывода контрольных осциллограмм.

Эта модель также содержит ряд субмоделей. При желании их можно просмотреть. Одна из них, содержащая блок SF-диаграммы, представлена на рис. 13.34. Здесь также можно найти множество интересных примеров задания описаний различных объектов SF-диаграммы, в частности ее состояний и управляемых событиями переходов.

Этим примером мы и 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. Возможно, это станет первым шагом на пути освоения им техники событийного моделирования. Применение пакета Stateflow совместно с мастерской реального времени Real Time открывает возможности применения системы MATLAB + Simulink для управления вполне реальными системами и устройствами, работающими в реальном масштабе времени. Однако такое применение выходит далеко за рамки материала данного самоучителя.

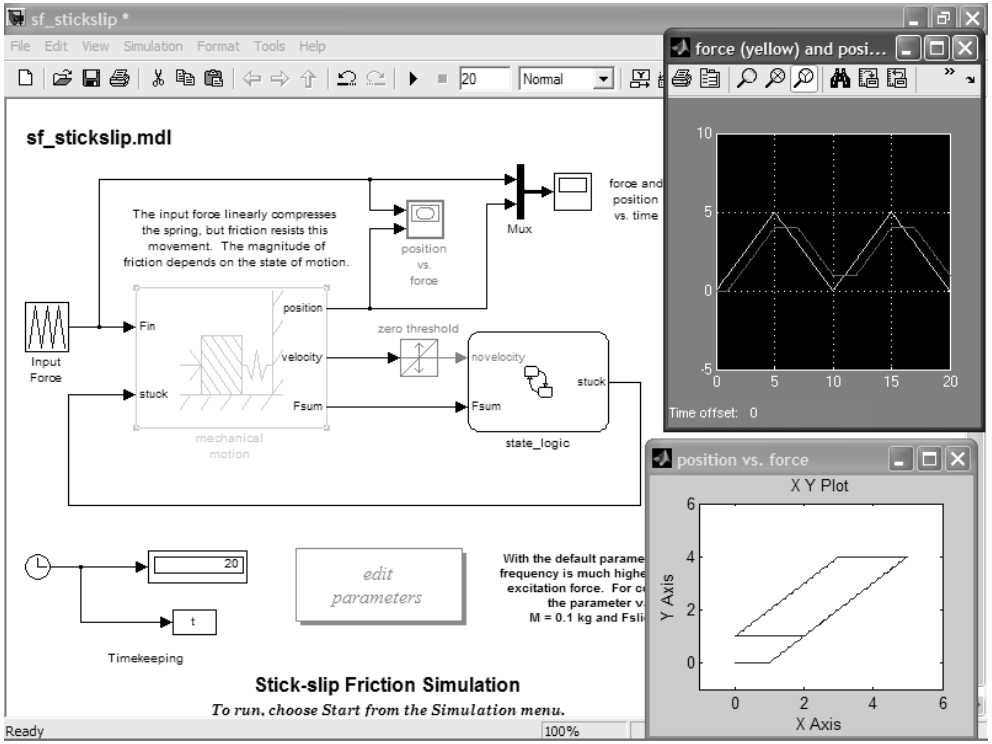


Рис. 13.28. Модель скользящего с трением бруска

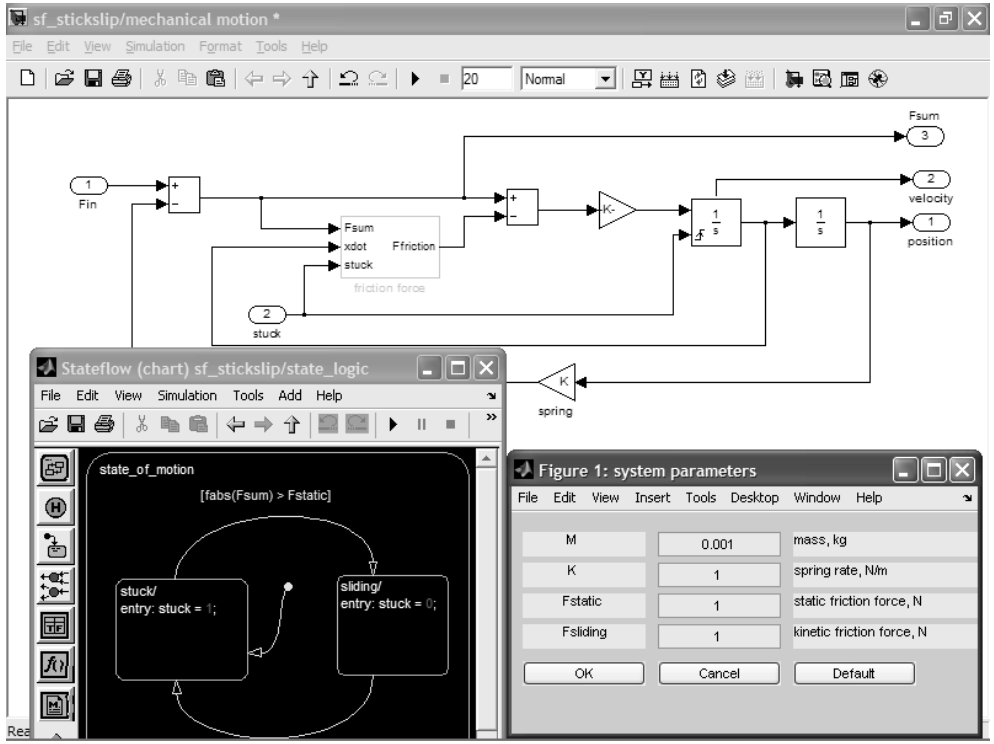


Рис. 13.29. Подсистема Mechanica Motion, SF-диаграмма модели скользящего с трением бруска и окно редактирования параметров

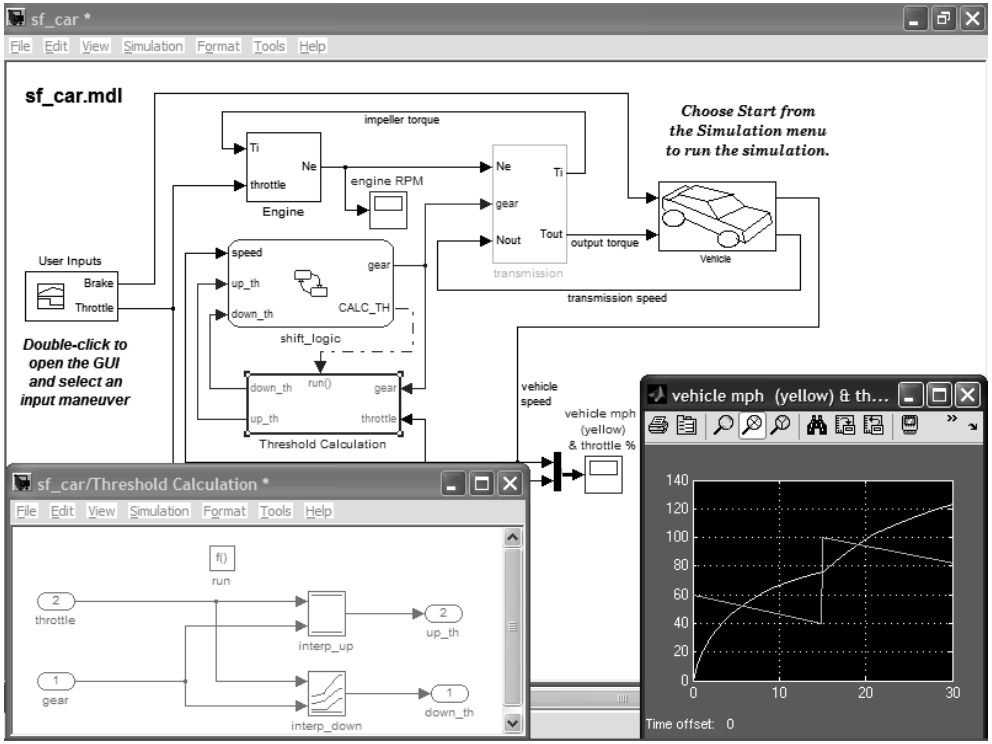


Рис. 13.30. Модель системы автоматической трансмиссии автомобиля

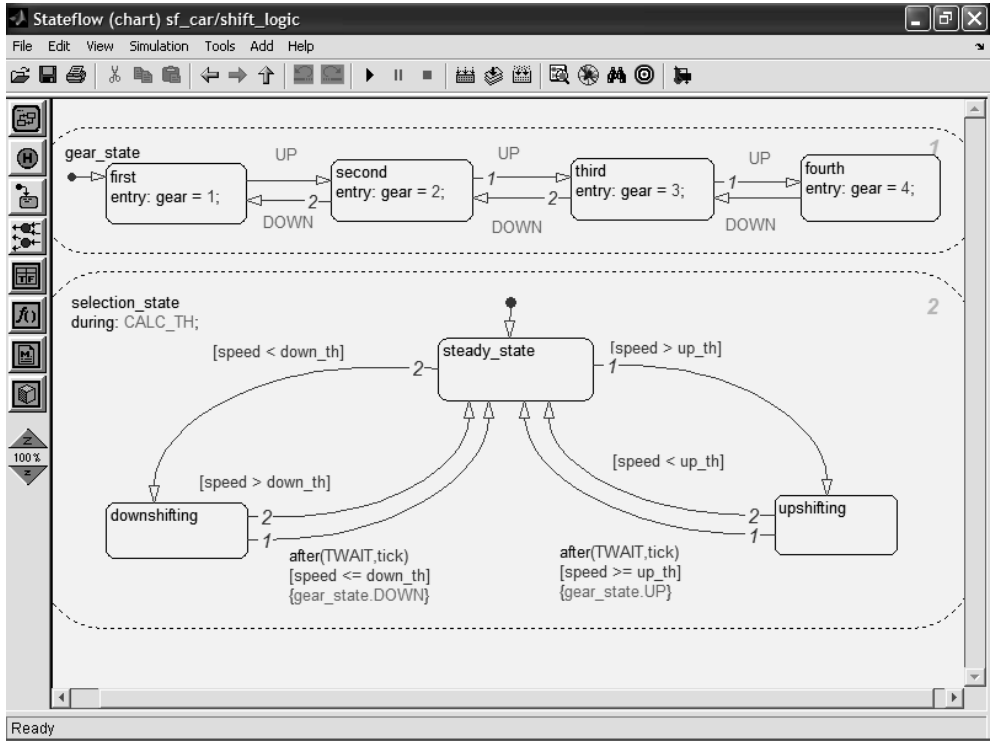


Рис. 13.31. SF-диаграмма модели рис. 13.30

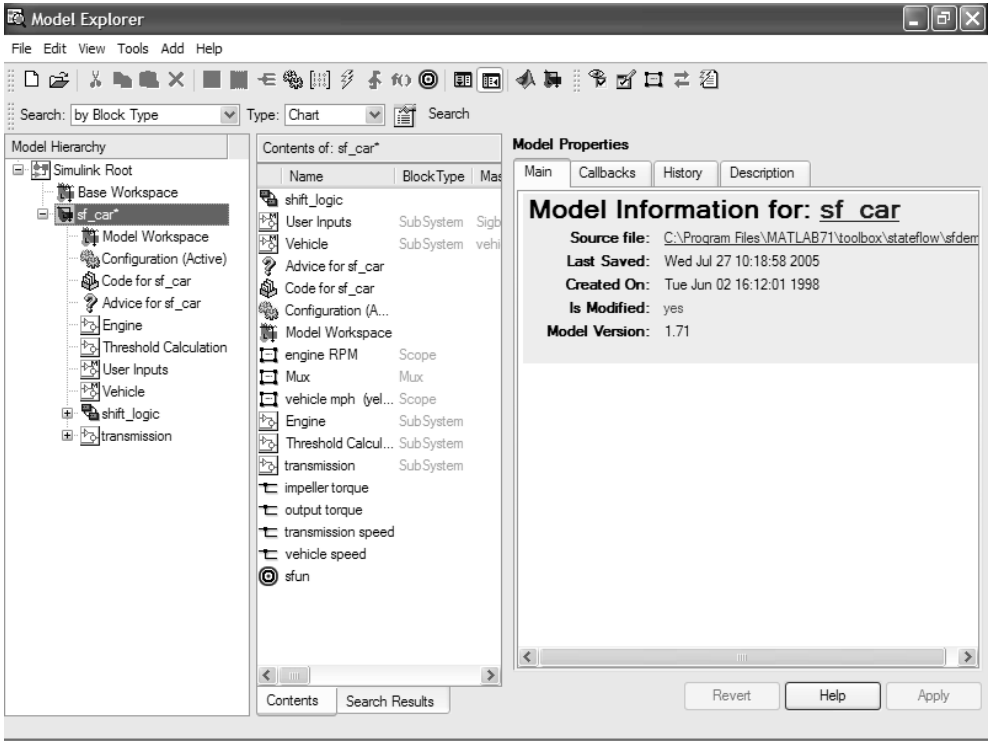


Рис. 13.32. Работа с обозревателем SF-диаграмм

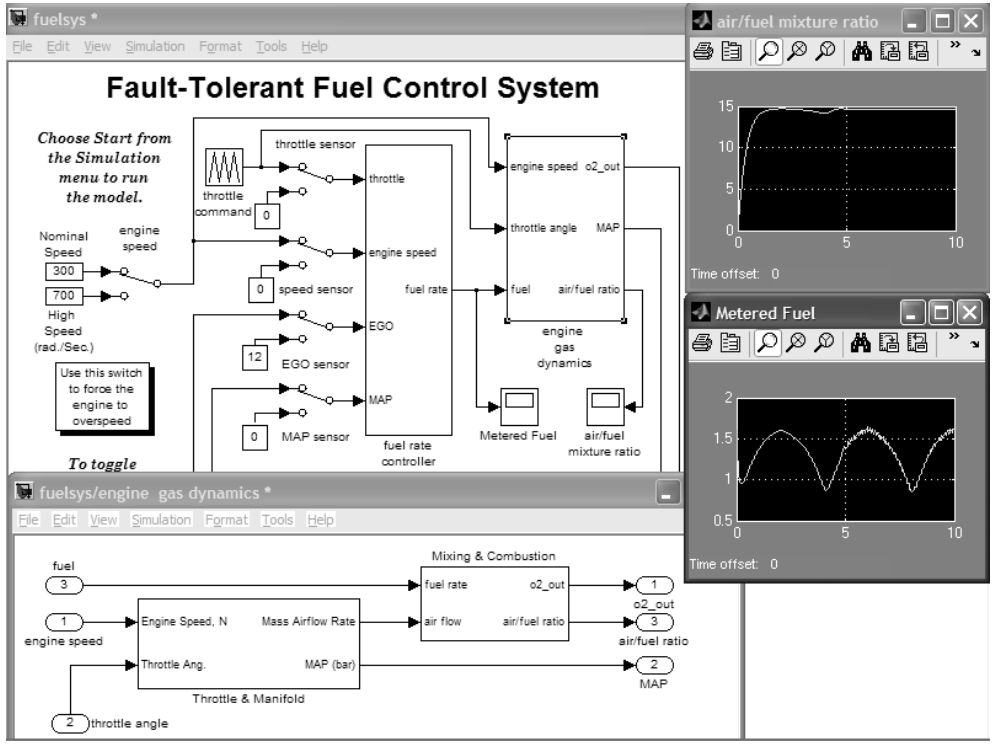


Рис. 13.33. Модель системы контроля топлива

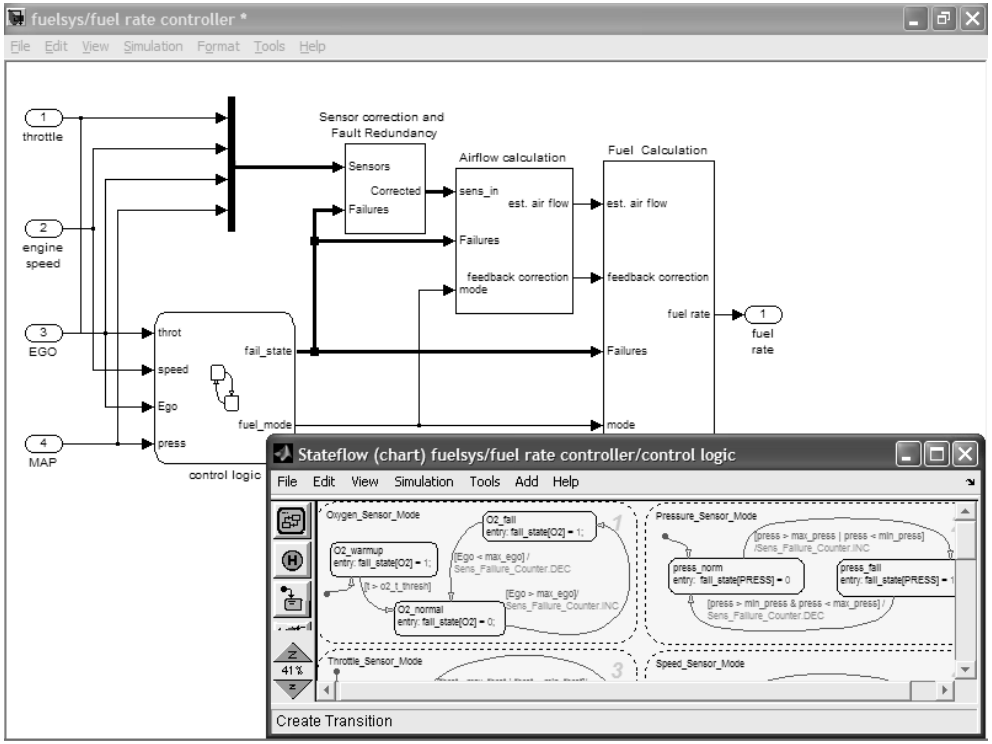


Рис. 13.34. Субмодель с SF-диаграммой системы контроля топлива

Урок 14

Моделирование устройств обработки сигналов и изображений

В этом уроке мы познакомимся с четырьмя важными пакетами расширения, входящими в инструментальный ящик Simulink – Blockset 6.5 (это последняя версия на момент подготовки данной книги). Описаны следующие пакеты расширения: **Signal Processing** (пакет моделирования по обработке сигналов, **RF** (пакет моделирования радиочастотных цепей), **Communication** (пакет моделирования коммуникационных систем) и **Video and Image Processing** (моделирование видеопотоков и видеоизображений). Все эти пакеты являются мощными расширениями системы Simulink, и знакомство с ними носит обзорный характер, впрочем, вполне достаточный для начала практической работы с этими пакетами.

14.1. Пакет расширения Signal Processing Blockset 6.5

14.1.1. Назначение пакета расширения Signal Processing Blockset 6.5

Пакет **Signal Processing Blockset 6.5** (SPB) является новейшей реализацией пакета расширения по обработке сигналов, предназначенного для работы с Simulink 6.6 (MATLAB R2007a). В старых версиях MATLAB он назывался **Digital Signal Processing Blockset** (сокращенно DSP) и вобрал в себя все возможности пакета расширения DSP с некоторыми расширениями. Применение этих пакетов целесообразно только для тех читателей, которые теоретически и практически знакомы с техникой цифровой обработки сигналов.

Для получения информации обо всех средствах пакета следует в командной строке MATLAB выполнить команду:

```
>> help dspblks\dspblks
  Signal Processing Blockset
  Version 6.5 (R2007a) 01-Feb-2007
  What's New (new features, bug fixes, and changes in this version)
    To display the Release Notes for Version 6.5, type
    "whatsnew" at the MATLAB command prompt.
  Signal Processing Blockset Functions
    dsplib — Open the Signal Processing Blockset library
    dsp_links — Display and return block library link information
    dspstartup — Configure Simulink for signal processing systems
    rebuffer_delay — Compute the delay introduced by the Buffer
block
   For more information, refer to the Signal Processing Blockset
documentation.  Signal Processing Blockset
```


14.1.2. Состав блоков библиотеки пакета Signal Processing Blockset 6.5

Доступ к библиотеке пакета можно получить, набрав имя одного из приведенных выше M-файлов или открыв браузер библиотек (рис. 2.21) и затем окно нужной библиотеки. Основное окно библиотеки пакета **Signal Processing Blockset 6.5** представлено на рис. 14.1. Там же показаны окна разделов библиотеки по источникам и получателям сигналов и по их фильтрации.

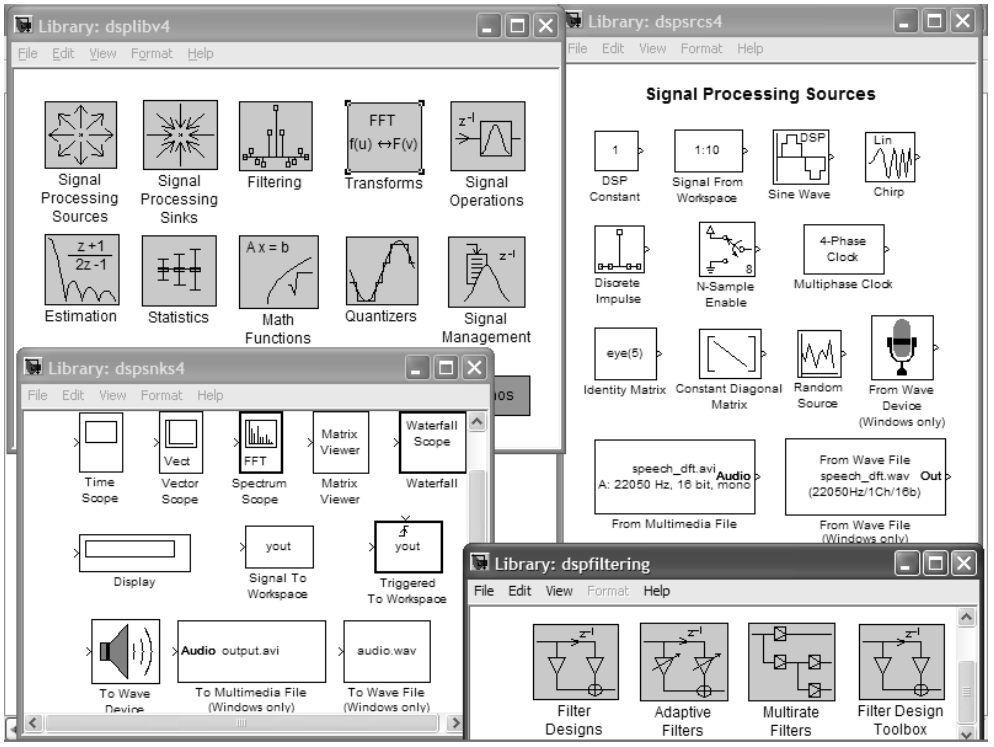


*Рис. 14.1. Окно библиотеки пакета **Signal Processing Blockset 6.5** и окна разделов источников и получателей сигналов и фильтрации*

Библиотека пакетов **DPB** и **DSP** содержит следующие разделы:

- **Signal Processing Sources** – источники сигналов;
- **Signal Processing 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.

На рис. 14.2 показан набор блоков следующих разделов библиотеки: преобразований, операций с сигналами и квантования.

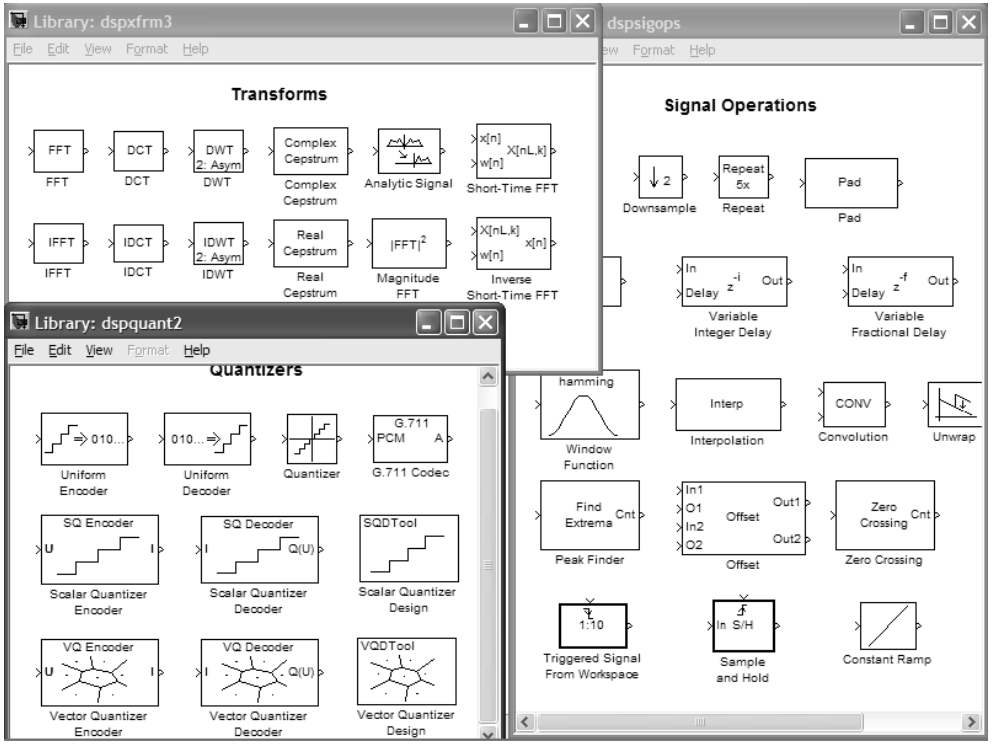


*Рис. 14.2. Окна разделов библиотеки пакета **Signal Processing Blockset 6.5**: преобразований, операций с сигналами и квантования*

Окна с разделами библиотеки по математическим и матричным операциям (набор 1), а также по операциям с полиномами показаны на рис. 14.3.

На рис. 14.4 показаны окна разделов библиотеки пакета **Signal Processing Blockset 6.5** с матричными операциями (набор 2), операциями факторизации и решения систем линейных уравнений.

Как видно по рис. 14.1–14.4, набор блоков пакета расширения **Signal Processing Blockset 6.5** весьма представителен и может решать огромное число задач моделирования устройств и систем обработки сигналов.

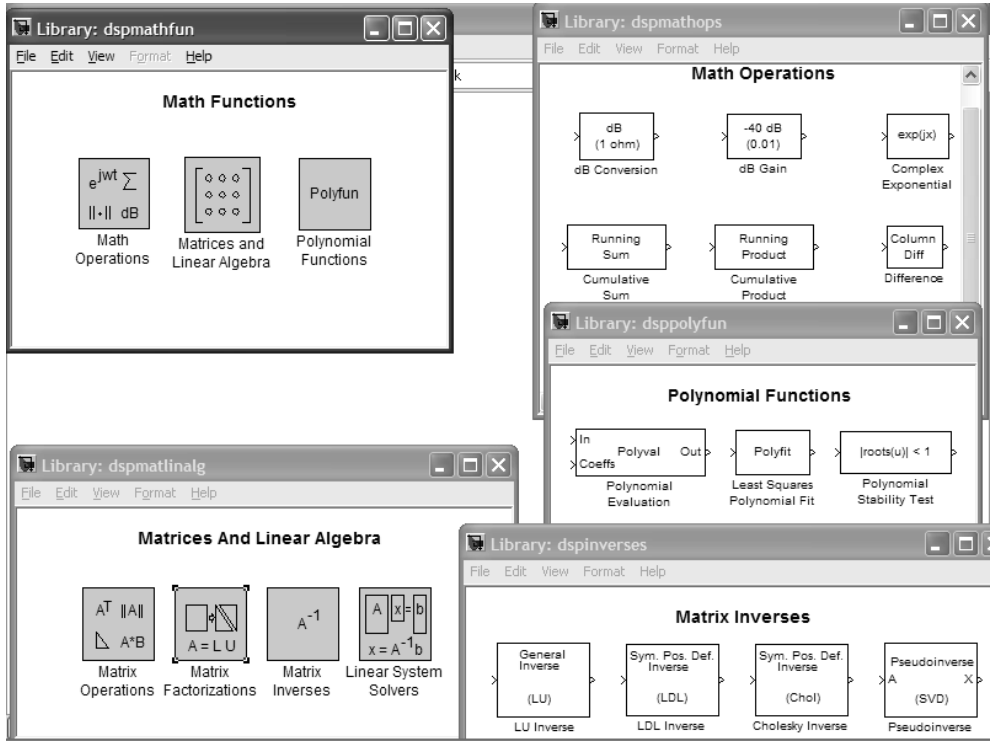


*Рис. 14.3. Окна разделов библиотеки пакета **Signal Processing Blockset 6.5**: с математическими и матричными операциями (набор 1) и операции с полиномами*

14.1.3. Работа с источниками и получателями сигналов

Сигналы обычно контролируются с помощью получателей сигналов, окно с которыми дано в разделе библиотеки **Sinks**. Здесь опять видны хорошо знакомые нам устройства, например дисплей **Display** или осциллограф – он теперь называется **Time Scope**.

Рисунок 14.5 показывает применение наиболее распространенных источников сигналов и их получателей. Здесь даны 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.

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

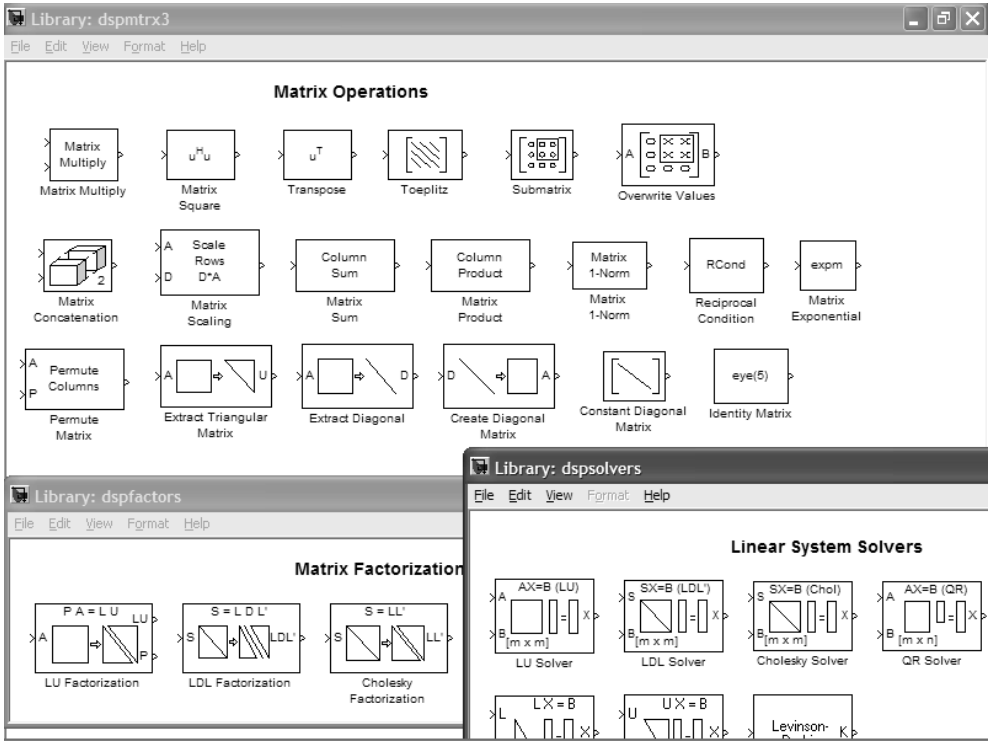


*Рис. 14.4. Окна разделов библиотеки пакета **Signal Processing Blockset 6.5**: с матричными операциями (набор 2), операциями факторизации и решения систем линейных уравнений*

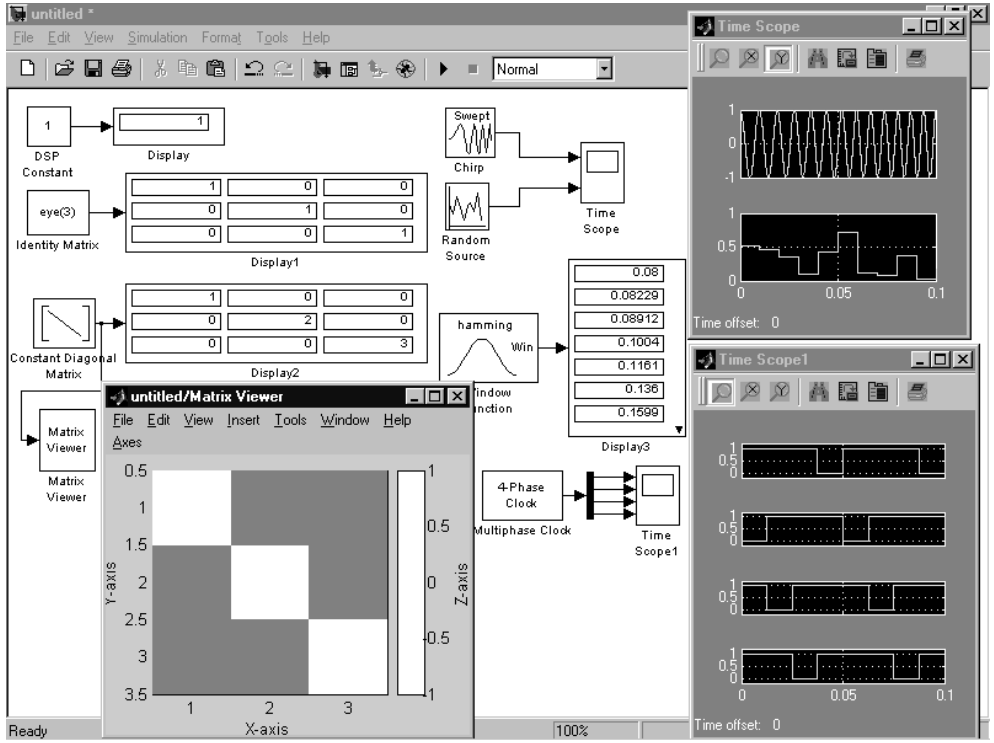


Рис. 14.5. Примеры работы с источниками и получателями сигналов

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.

14.1.4. Работа с блоками математических операций

Пакет **Signal Processing Blockset** имеет весьма обширные возможности в проведении математических операций. Они сосредоточены в разделе его библиотеки с названием **Math Function**. Примеры выполнения математических операций представлены на рис. 14.6.

Эти операции в особых комментариях не нуждаются. Надо просто внимательно просмотреть фиксируемые на выходе блоков значения сигналов.

14.1.5. Типовые матричные операции

Для обращения матриц пакет имеет четыре блока. Примеры применения их даны на рис. 14.6 в правом верхнем углу. Обратите внимание, что смысл операции обращения в разных методах разный, а потому отличаются и результаты.

Другие матричные операции представлены весьма внушительным набором блоков (рис. 14.3 и 14.4). Ввиду их общеизвестности комментировать эту группу операций мы не будем – названия блоков и вид их пиктограмм говорят сами за себя.

В правой части рис. 14.6 показан пример выделения диагонали матрицы с помощью блока **Extract Diagonal**. Ряд примеров на применение большинства блоков этого раздела представлен на рис. 14.7. Обратите внимание, что большинство операций выполняется над простой квадратной матрицей [1 2; 3 4]. Для регистра-

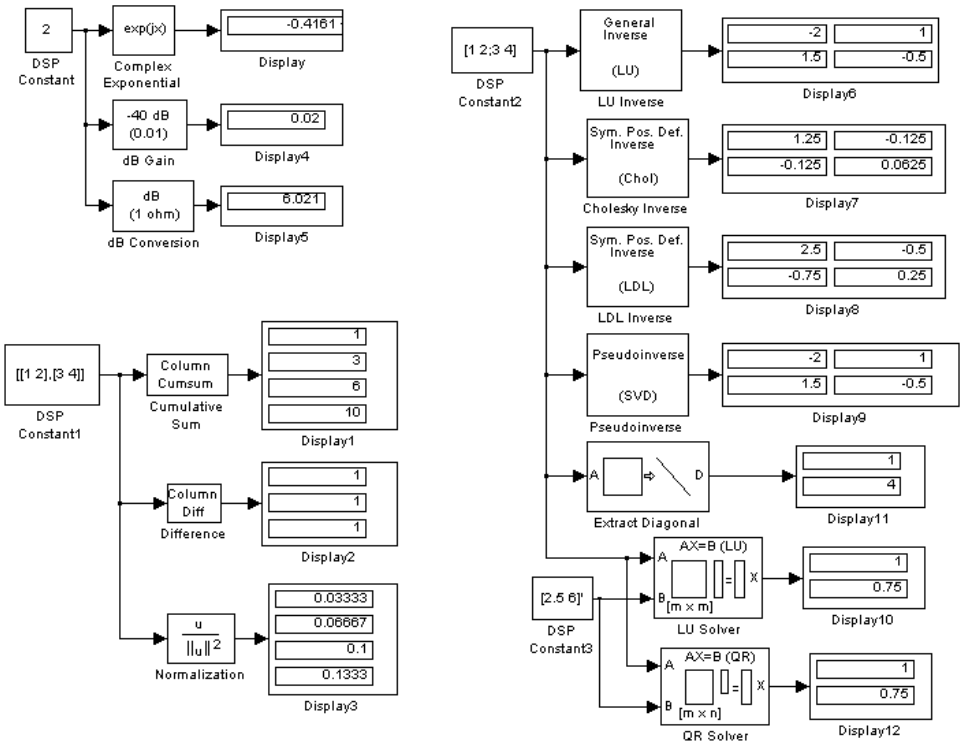


Рис. 14.6. Примеры выполнения математических операций

ции результатов операций использован цифровой индикатор – дисплей, пиктограмма которого растягивается до размера, достаточного для наблюдения всех результатов.

Важной задачей линейной алгебры является решение систем линейных уравнений вида $AX = B$, где A – квадратная матрица коэффициентов 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. 14.7 в верхнем правом углу окна), пакет **DSP** предоставляет четыре специальных блока для решения систем линейных уравнений. В правой части рис. 14.6 даны два примера на решение системы линейных уравнений с помощью этих блоков:

$x_1 + 2x_2 = 2{,}5$
$3x_1 + 4x_2 = 6$

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 в пакете **DSP** реализована пятью блоками. Пример LU-разложения дан на рис. 14.7.

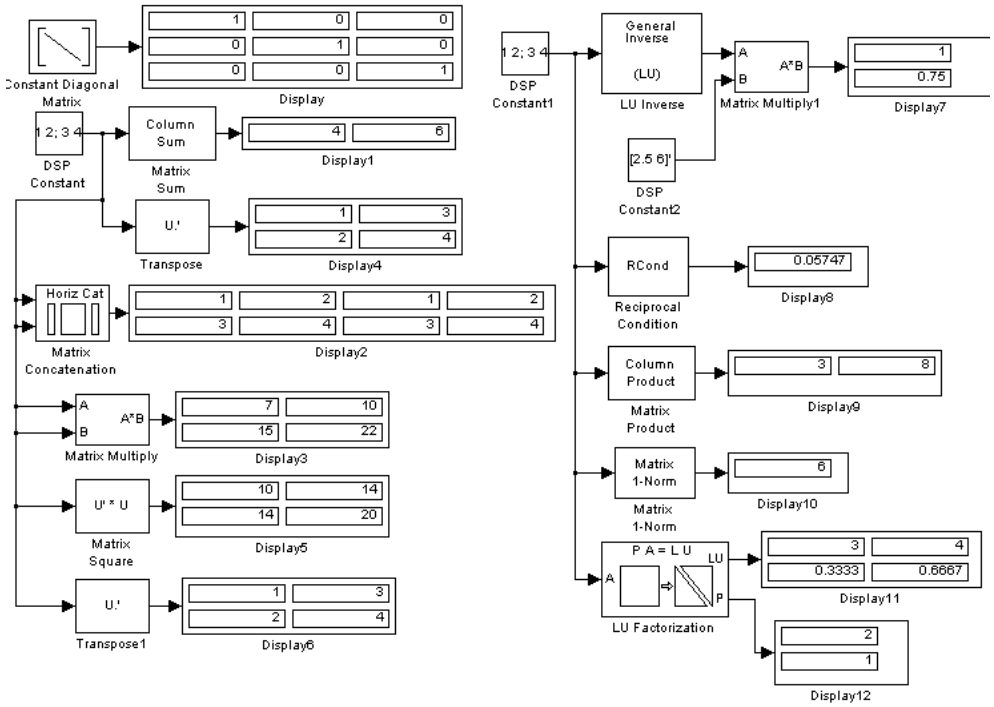


Рис. 14.7. Примеры применения блоков типовых матричных операций

14.1.6. 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, которые обеспечивают: вычисление значений полинома, уточнение его коэффициентов по минимуму среднеквадратической погрешности в заданных точках и определение устойчивости полинома по его корням (см. примеры на рис. 14.8).

Обратите внимание на блок вычисления значения полинома **Polynomial Evalution**. Он использован трижды (при вычислении значения полинома для эле-

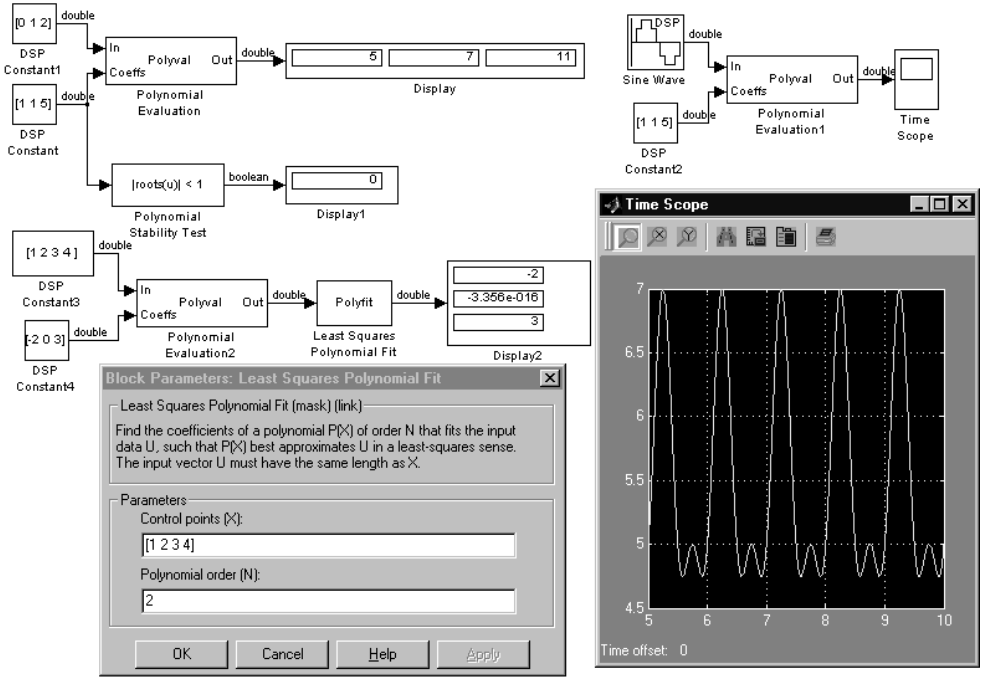


Рис. 14.8. Операции с полиномами

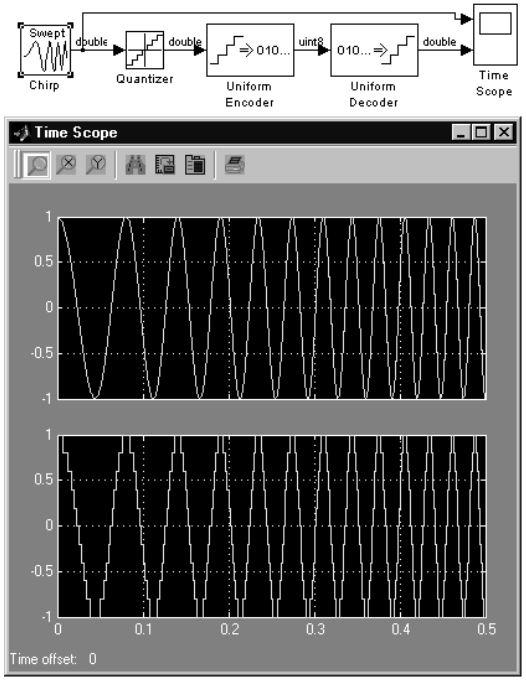


Рис. 14.9. Пример квантования сигнала переменной частоты

ментов 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. 14.8.

14.1.7. Квантование сигналов

В состав раздела **Quantizers (Квантователи)** входят три блока: квантователь **Quantizer**, кодирующий блок **Uniform Encoder** и декодирующий блок **Uniform Decoder**. На рис. 14.9 дана простая и наглядная модель канала для цифрового кодирования и декодирования синусоидального сигнала с нарастающей во времени частотой. 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 декодирующего 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).

14.1.8. Управление сигналами

Средства управления сингалами сосредоточены в разделе **Signal Managements** библиотеки – рис. 14.10.

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 (признаков) сигнала.

14.1.9. Организация буфера, очереди и стека

Один из важнейших блоков **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. 14.11).

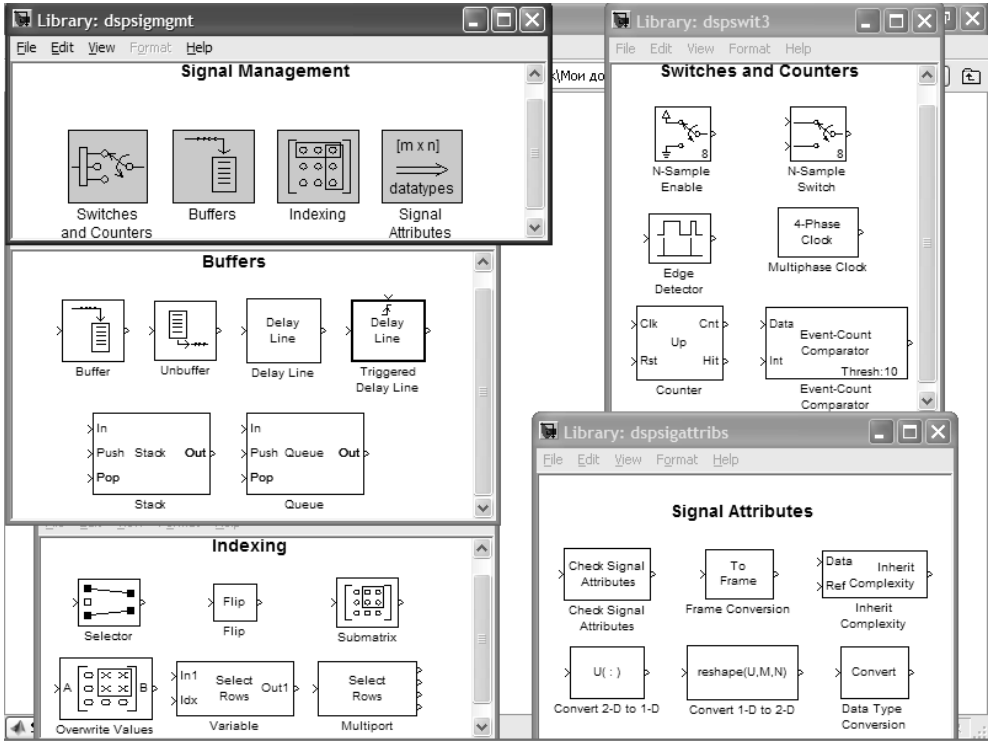


Рис. 14.10. Раздел библиотеки управления сигналами

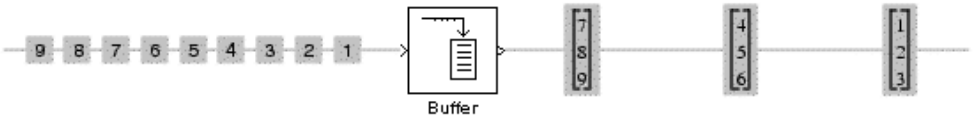


*Рис. 14.11. Работа блока **Buffer***

Буфер характеризуется тремя параметрами (в скобках дано значение параметра по умолчанию):

- **Buffer size** – размер буфера ($M = 64$), или 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 14.12 иллюстрирует работу буфера при входном сигнале в виде вектора.

*Рис. 14.12. Работа блока **Buffer** при векторном входном сигнале*

Рисунок 14.13 показывает работу буфера при $M = 3$ и $L = 1$, то есть при наличии перекрытия.

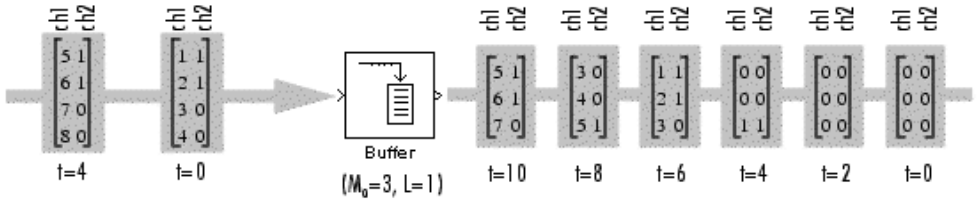


*Рис. 14.13. Работа блока **Buffer** при наличии перекрытия*

Если входной вектор содержит N элементов, а параметр **Buffer size** равен M, то, как видно по рис. 14.11, каждый фрейм будет матрицей размера $M \times N$, каждый элемент которой – скаляр, полученный по отдельному каналу $\mathbf{h}_i$ (**i** – номер канала). Если входной сигнал – матрица размера $m \times n$, то она преобразуется в вектор, содержащий mn элементов.

Блок **Unbuffer** собирает фреймы в один поток (рис. 14.14). Если фреймы разделены интервалом времени 1, то скалярные значения выходного сигнала будут идти с интервалом t/M (в нашем примере 1/3).

Этот блок имеет один параметр – начальное состояние **Initial condition** (0). Рисунок 14.15 показывает работу блока при четырехканальном входном сигнале, каждый фрейм которого – матрица размера 3×32.

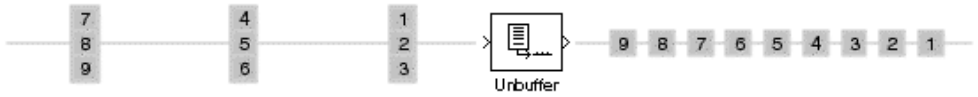


*Рис. 14.14. Работа блока **Unbuffer** при векторном входном сигнале*

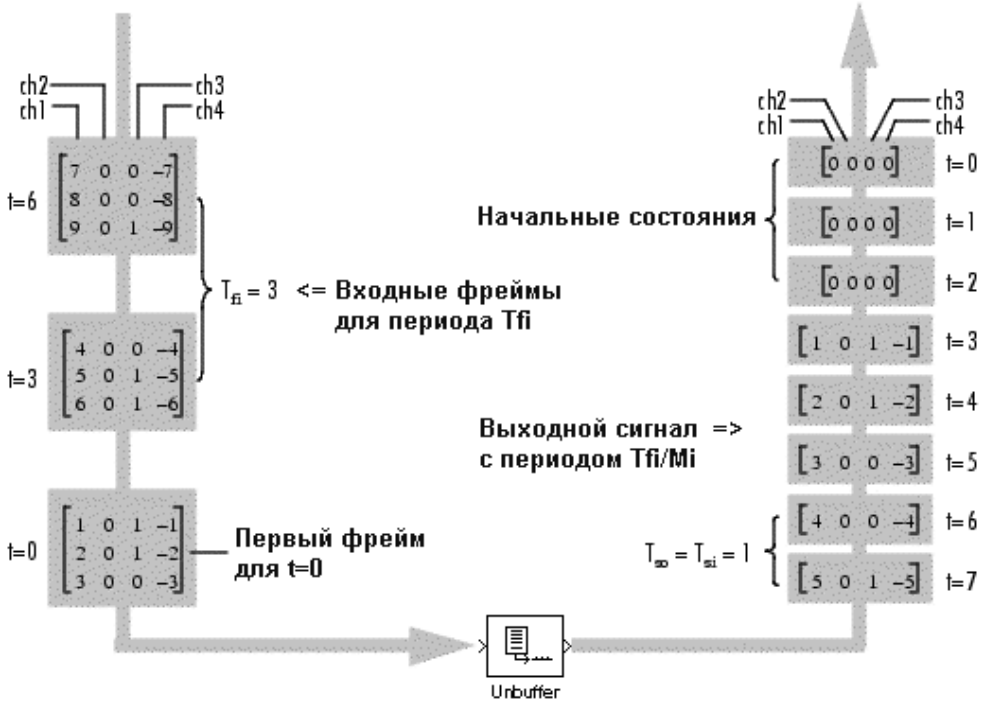


*Рис. 14.15. Работа блока **Unbuffer** при матричном входном сигнале*

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. 14.16.

Если на порт **Push** дан сигнал «Пустить в очередь», то очередное значение (вектор или матрицу) сигнала с 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**.

Блок **Queue** имеет следующие параметры:

- **Register size** – размер регистра, или максимальная длина очереди;

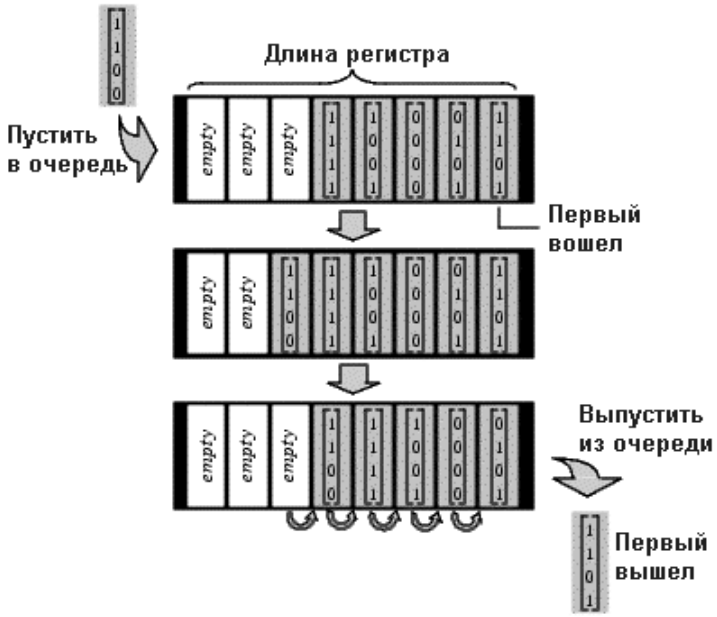


*Рис. 14.16. Организация очереди с помощью блока **Queue***

- **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**.

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. 14.17.

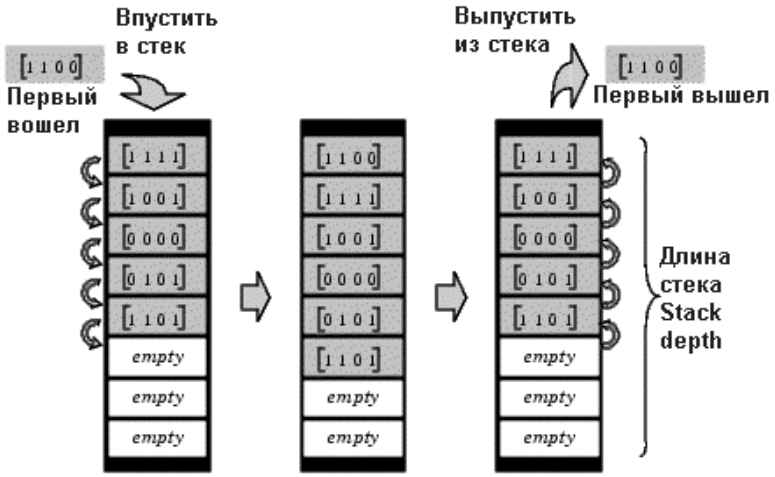


*Рис. 14.17. Диаграмма работы блока **Stack***

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).

14.1.10. Организация сдвигового регистра и линии задержки

Для организации сдвигового регистра в версии Simulink 3 предназначался блок **Shift Register**. В дальнейшем он был переименован в линию задержки **Delay Line**. Диаграмма работы блока представлена на рис. 14.18.

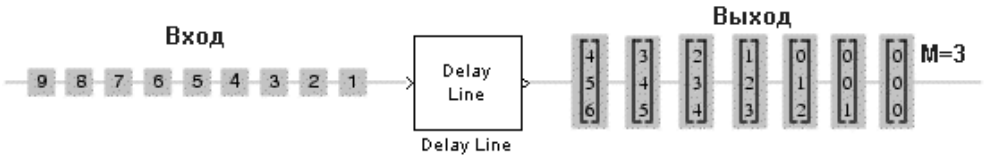


*Рис. 14.18. Работа блока **Delay Line***

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** 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, то сдвигающий регистр поочередно то работает, то нет. Таким образом, функционально он подобен сдвигающему регистру с триггером на входе. Блок имеет параметр **Trigger type**, определяющий тип управляющего сигнала: можно задать переключение режима работы регистра при положительном, отрицательном или при любом перепаде.

14.1.11. Подраздел Signal Attributes

Подраздел **Signal Attributes** раздела **Signal Managements** библиотеки имеет шесть блоков:

- **Check Signal Attributes** – проверка атрибутов сигнала;
- **Convert 2-D to 1-D** – преобразование 2D-сигнала в сигнал 1D;
- **Convert 1-D to 2-D** – преобразование 1D-сигнала в сигнал 2D;
- **Continuous Copy** – создает продолженную копию данных;
- **Frame Status conversion** – преобразование статуса фреймов;
- **Inherit Complexity** – наследование комплексности данных.

В данном случае преобразования осуществляются над одномерными сигналами типа 1D и двумерными 2D (то есть представленными, соответственно, векторами и матрицами). На рис. 14.19 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.

14.1.12. Переключатели и счетчики

Подраздел **Switches and Counters** раздела **Signal Managements** представлен блоками переключателей:

- **N-Sample Enable** – блок включения неактивного сигнала; задает в течение заданного времени неактивное состояние сигнала, а затем – активное;
- **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 нуль, 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.

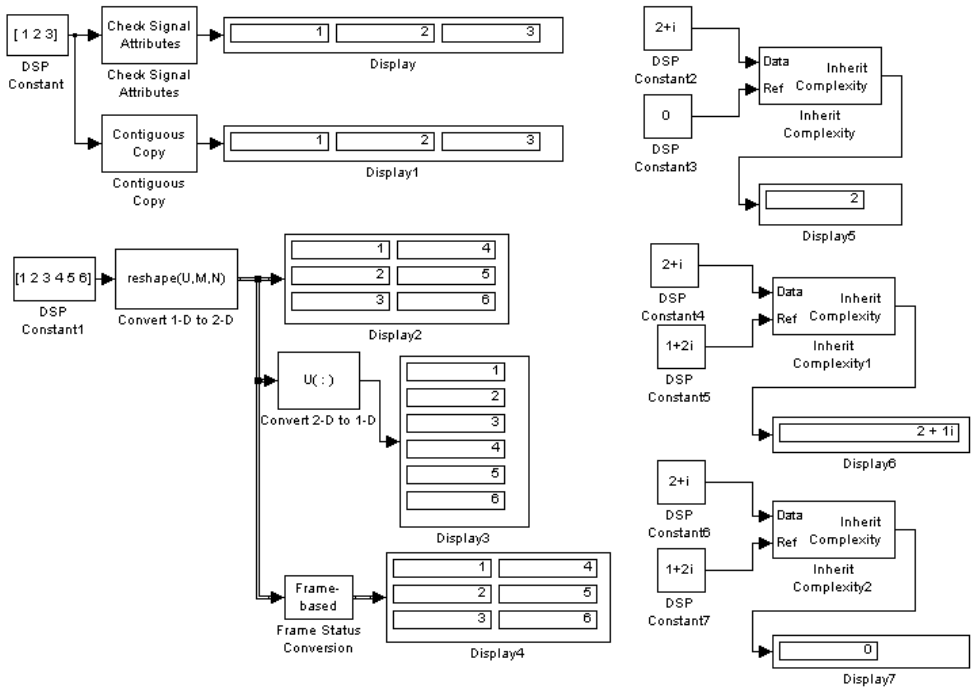


Рис. 14.19. Примеры работы блоков ***Signal Attributes***

Работу блоков **N-Sample Enable**, **N-Sample Switch** и **Edge Detector** поясняет рис. 14.20. Он содержит осциллограммы работы блоков и окна их параметров. Параметр **N** задает число тактов, по истечении которых меняется состояние переключателей.

Типовое 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.

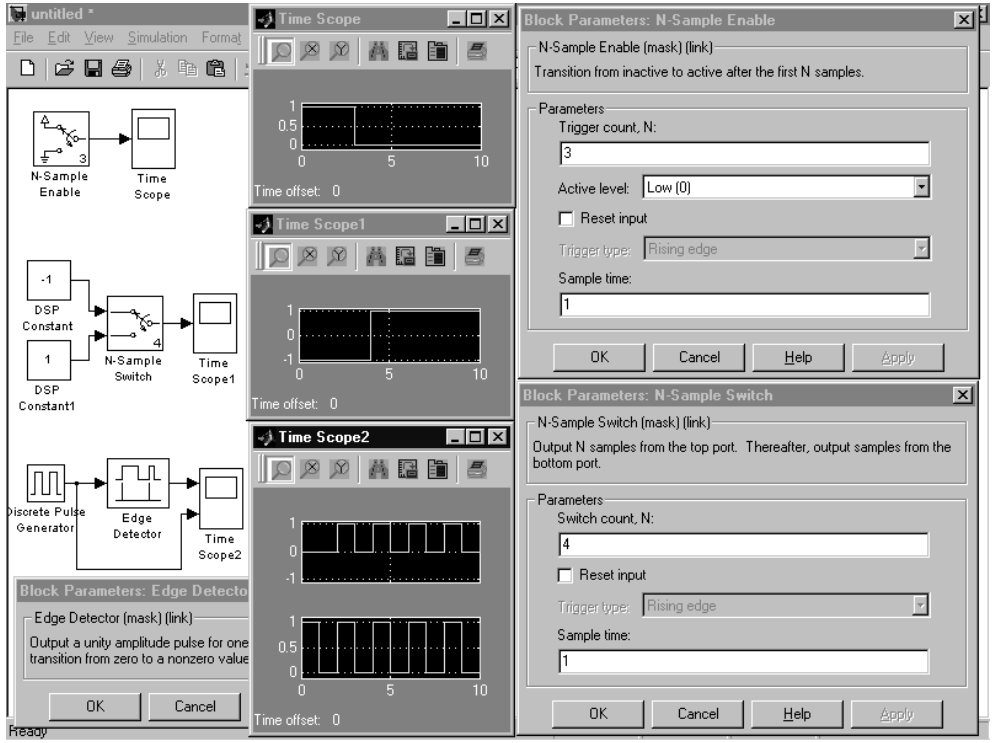


Рис. 14.20. Работа переключателей подраздела ***Switches and Counters***

Работу счетных блоков **Counter** и **Event-Count Comparator** поясняет рис. 14.21. На нем также показаны окна параметров этих блоков.

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. Осциллограммы работы этих блоков представлены на рис. 14.22.

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

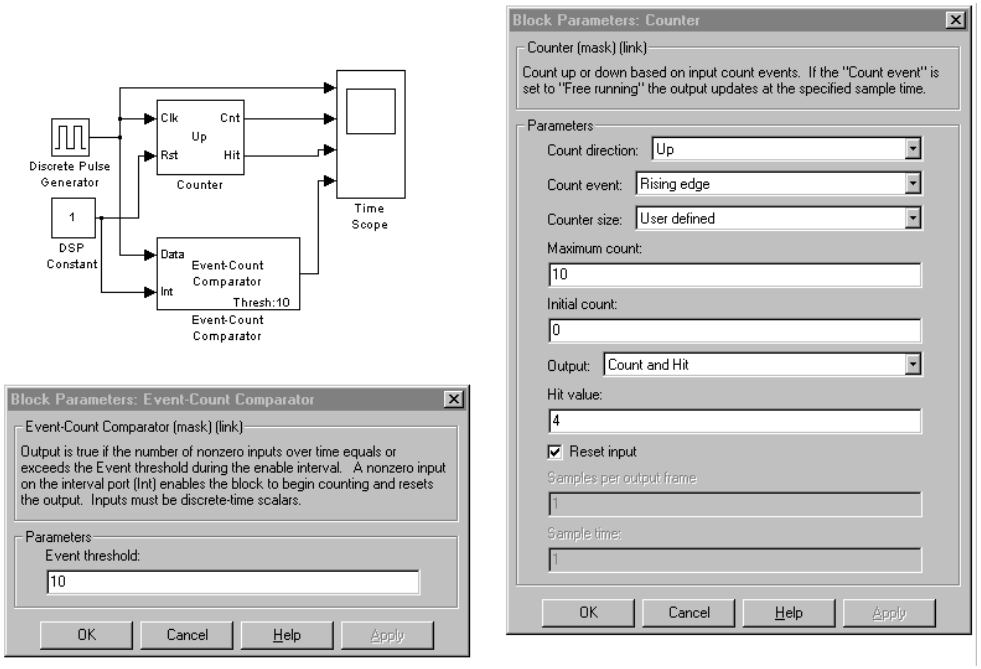


*Рис. 14.21. Блоки счетчиков подраздела **Switches and Counters***

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**.

14.1.13. Обработка сигналов (раздел Signal Operations)

Раздел **Signal Operations** библиотеки пакета представлен многими блоками. Они осуществляют различные операции с сигналами: увеличение числа отсчетов сигналов Upsample, уменьшение числа отсчетов Downsample, повторение сигнала

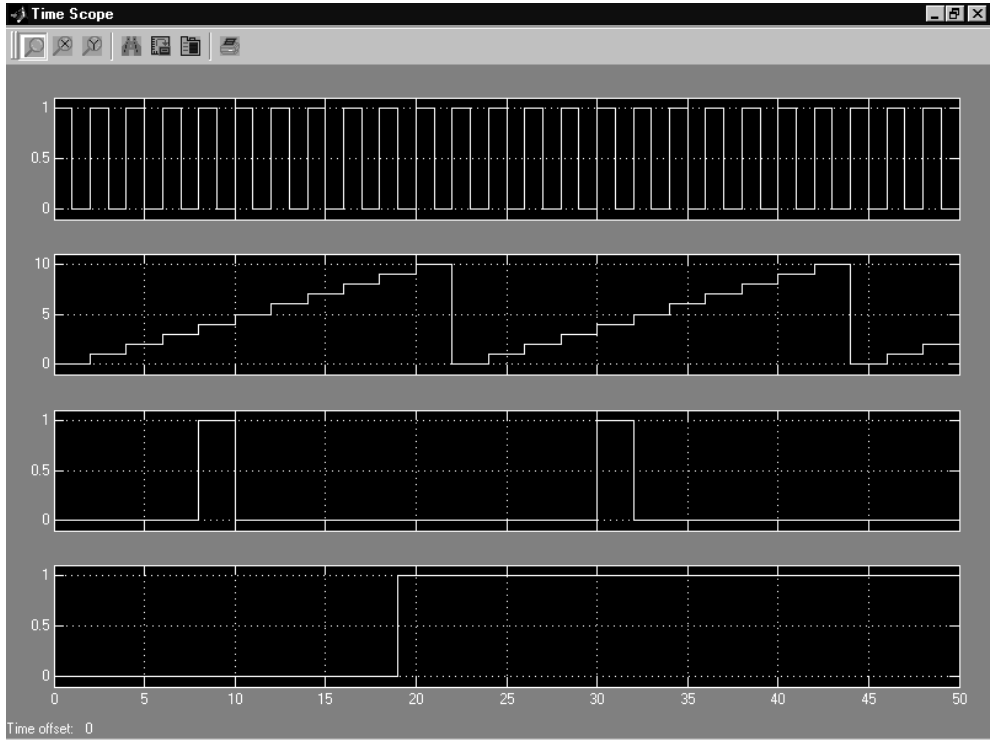


Рис. 14.22. Осциллограммы счетчиков

Repeat, задержки различного вида, задания окон Windows Function, свертки Convolution и др. Мы рассмотрим лишь основные из этих блоков.

Функции ряда блоков этого раздела хорошо иллюстрируются временными диаграммами их работы, представленными на рис. 14.23.

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**.

14.1.14. Раздел оценки блоков – DSP Estimation

Раздел библиотеки **Estimation** (блоки оценки) представлен основным разделом и тремя подразделами (рис. 14.24).

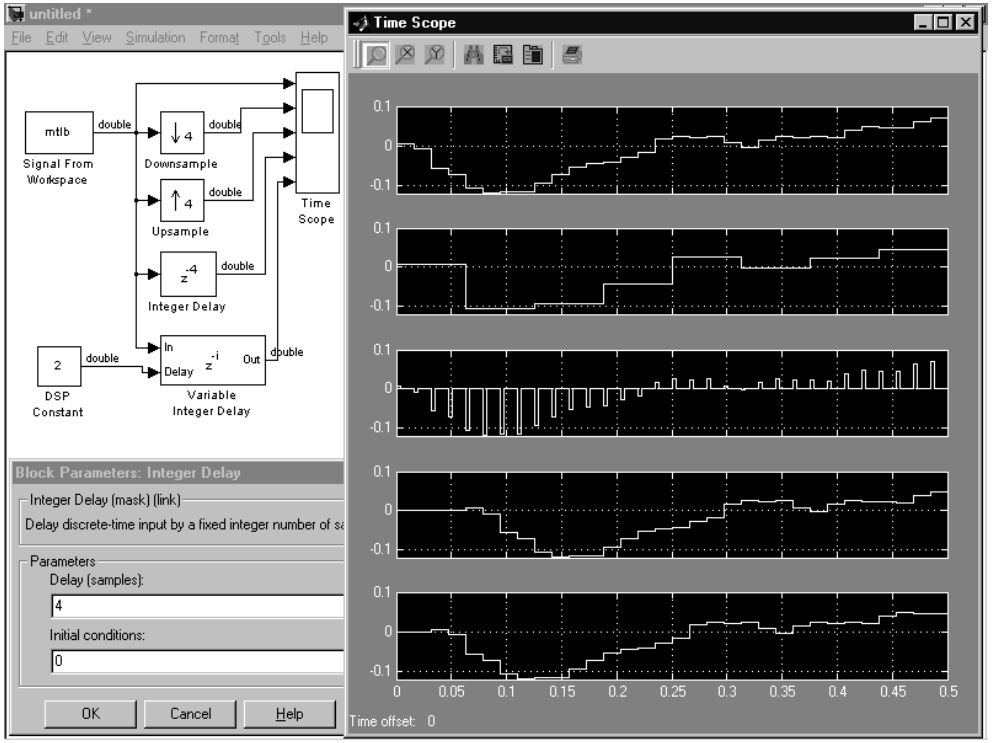


*Рис. 14.23. Примеры работы некоторых блоков раздела **Signal Operations***

Окно подраздела линейного предсказания **Linear Prediction** содержит единственный блок **Autocorrelation LPT**. Он вычисляет параметры автокорреляции для матрицы (в общем случае *M*×*N*) или вектора. 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.

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.

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. 14.25.

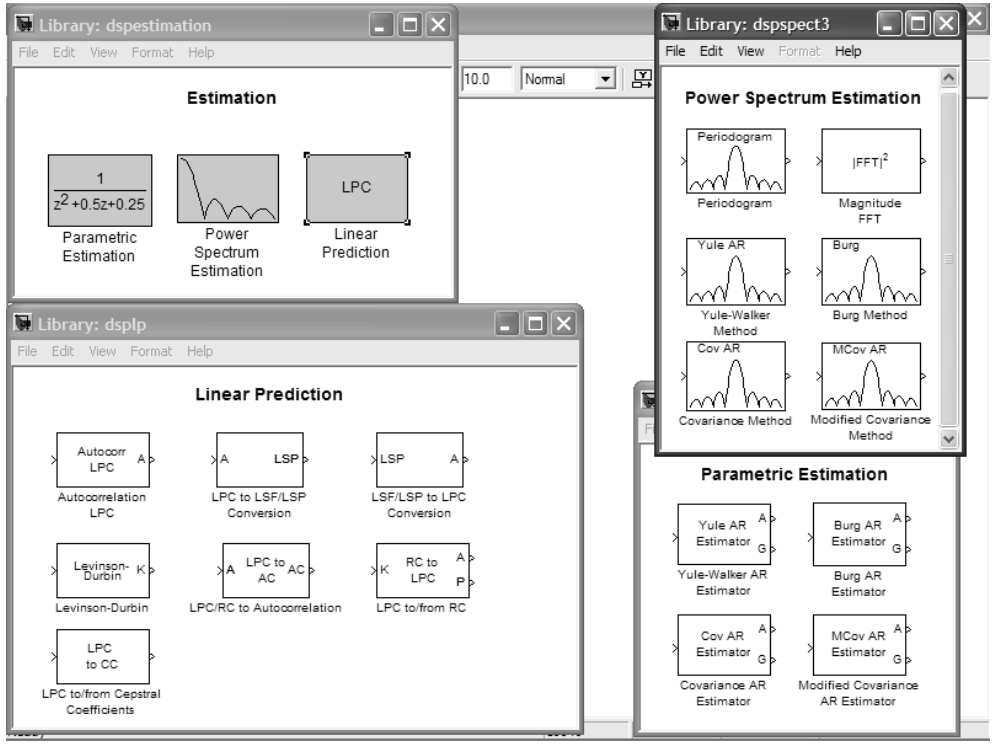


*Рис. 14.24. Окно раздела **Estimation***

К важнейшим параметрам этих блоков относятся длина (число полос) спектрограммы и параметр наследования размерности входных данных.

14.1.15. Преобразования сигналов (раздел Transforms)

Раздел **Transforms** содержит 10 блоков (рис. 14.2). В этом разделе представлены блоки прямого и обратного преобразований Фурье (FFT) и дискретного косинусного преобразования (DCT), а также блоки связанных с ними операций.

Блоки **FFT** и **IFFT** реализуют стандартное прямое и обратное преобразования Фурье для входной последовательности данных, число элементов которой должно быть целой степенью числа 2. 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).

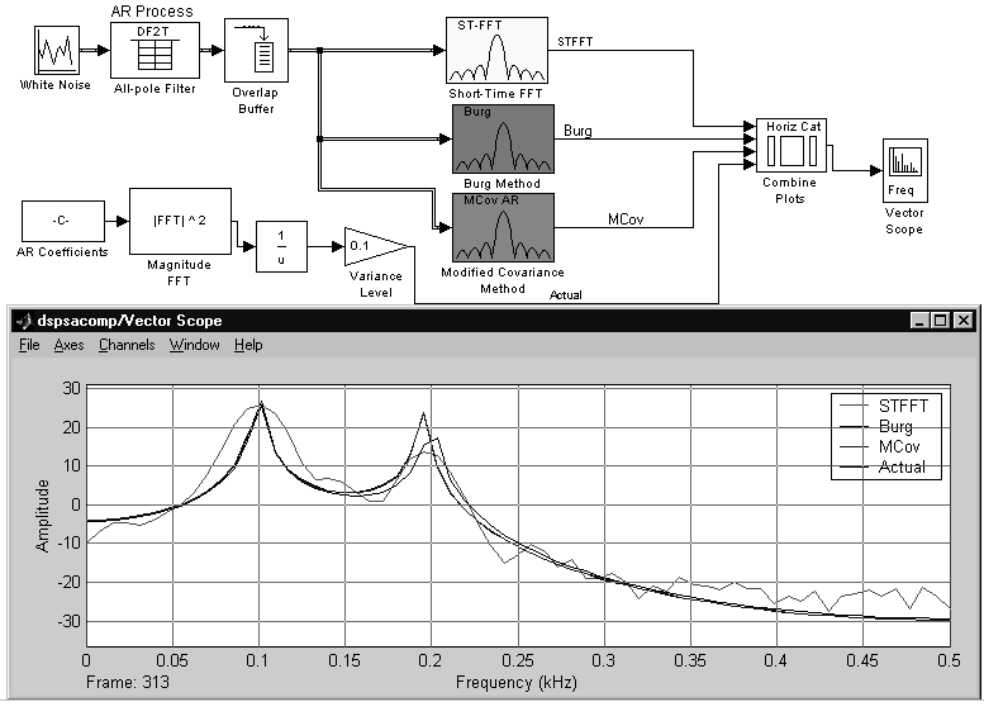


Рис. 14.25. Сравнение анализаторов спектра разного типа

Прямое и обратное дискретные косинусные преобразования реализуются блоками **DCT** и **IDCT**. Эти блоки также не имеют параметров – их окна установки параметров являются чисто информационными.

Блок **Analytic Signal** служит для преобразования каждого канала в общем случае матрицы размера $M \times N$ в комплексный аналитический сигнал вида $y = u + j H\{u\}$, где $H\{u\}$ – преобразование Гильберта. Единственным параметром этого блока является порядок фильтра **Filter order**.

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 рис. 14.26 показано применение некоторых из этих блоков раздела **Transform**. На выходе создается матрица с вещественными элементами размера $Mo \times N$, где Mo задается по данным FFT-преобразования.

В каждом блоке всего два параметра – один `Inherit FFT length from input port dimension` задает выбор длины вектора для FFT-преобразования как длину входного вектора, второй `FFT length` позволяет задать длину вектора произвольно (64 по умолчанию).

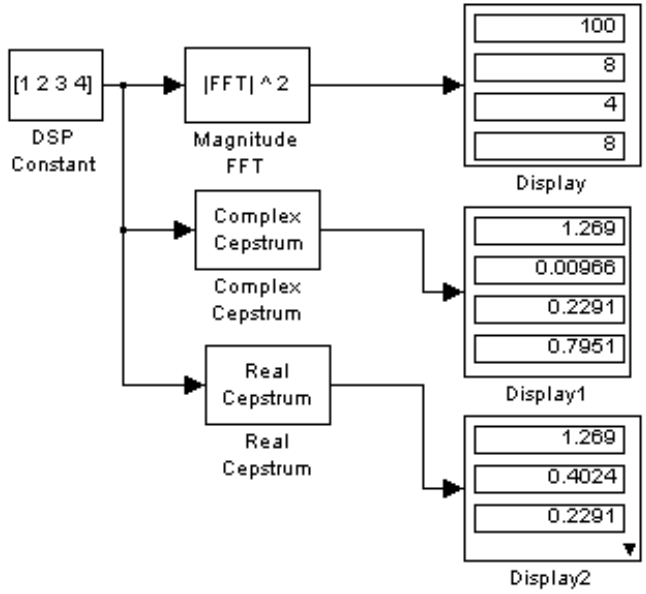


*Рис. 14.26. Примеры применения блоков **|IFFT|^2, Complex Cepstrum** и **Real Cepstrum***

14.1.16. Статистическая обработка данных (раздел DSP Statistics)

Окно с блоками раздела **DSP Statistics** изображено на рис. 14.27. В нем представлены 12 блоков, выполняющих типовые статистические вычисления, лежащие в основе статистической обработки данных и сигналов.

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;

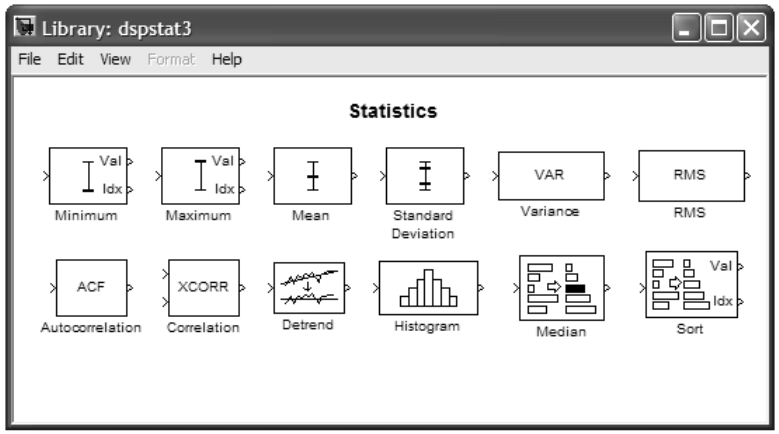


*Рис. 14.27. Окно раздела **DSP Statistics***

- **Median** – вычисление медианы;
- **Sort** – сортировка и вывод данных с их индексами.

Ряд блоков раздела **DSP Statistics** реализуют простейшие статистические вычисления. Примеры их применения представлены на рис. 14.28.

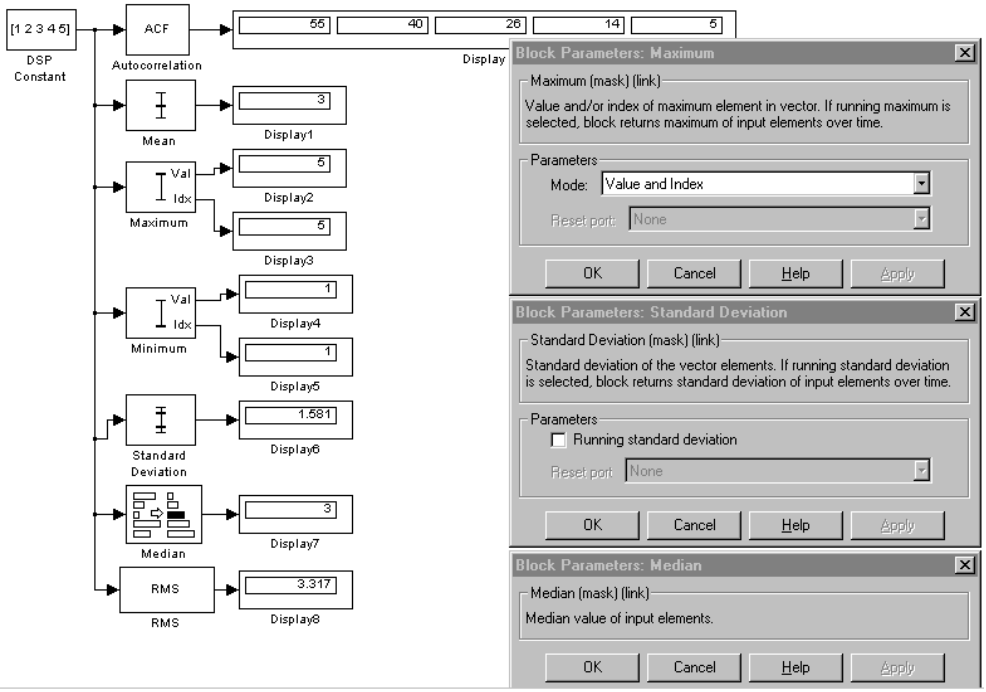


Рис. 14.28. Примеры простейших статистических операций

Ввиду очевидности этих вычислений их подробное описание (как и описание типовых параметров большинства блоков) опущено.

Блоками статистических преобразований являются те блоки, которые преобразуют входные данные множественного типа в выходные данные также множественного типа. Работа трех таких блоков (**Sort**, **Detrend** и **Histogram**) представлена на рис. 14.29. Там же показаны окна их параметров.

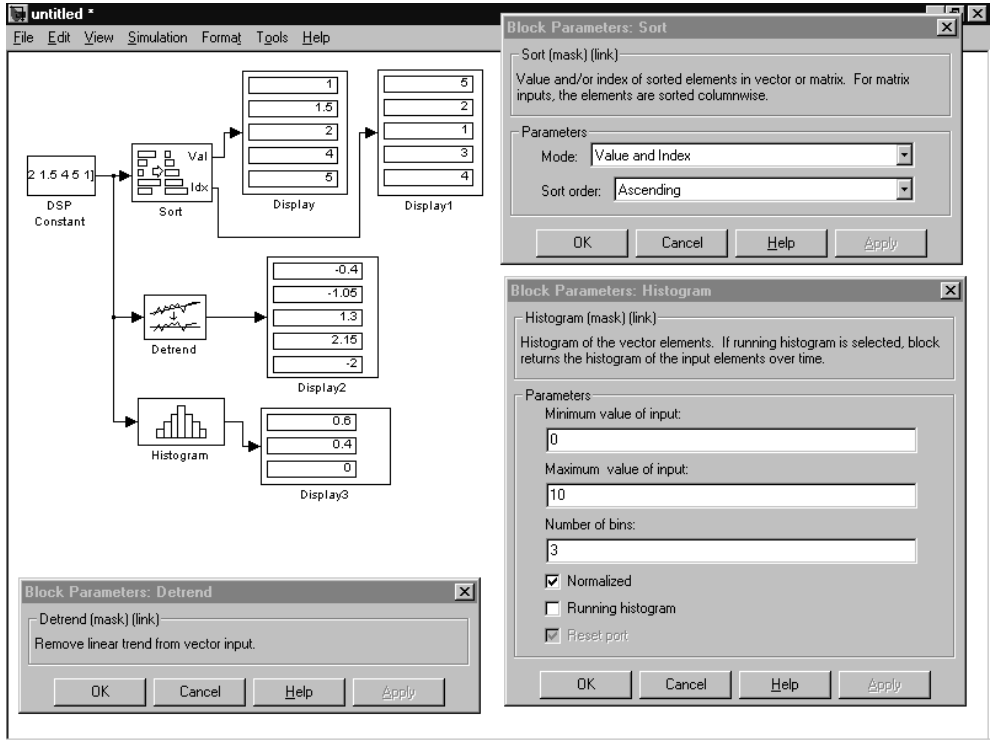


*Рис. 14.29. Применение блоков **Sort**, **Detrend** и **Histogramm***

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 **Correlation** вычисляет вектор кросс-корреляции для двух векторов входных данных. Параметров этот простой блок не имеет.

14.1.17. Фильтрация сигналов (раздел Filtering)

Раздел **Filtering** библиотеки пакета **DSP** является одним из самых крупных. Однако, поскольку моделирование фильтров представляет интерес для ограниченного круга читателей, мы рассмотрим этот раздел библиотеки обзорно. Он содержит четыре подраздела (рис. 14.1):

- **Filter Designs** – конструирование фильтров разного вида;
- **Filter Designs Toolbox** – структуры фильтров разного вида;
- **Adaptive Filter** – адаптивные фильтры;
- **Multirate Filter** – многополосные фильтры.

Подраздел **Filter Designs** содержит шесть блоков фильтров. Они позволяют реализовать фильтры типа FIR (с конечной импульсной характеристикой), IIR (с бесконечной импульсной характеристикой) и аналоговые фильтры. Подраздел **Filter Designs Toolbox** содержит ряд блоков фильтров, которые реализуют основные типы фильтрации сигналов. Подраздел **Adaptive Filter** имеет всего шесть блоков для построения адаптивных фильтров. Это фильтры Калмана, левой LMS, RMS и боковых полос. И еще девять блоков содержит подраздел **Multirate Filter**. Здесь представлены варианты FIR-фильтров (с конечной импульсной характеристикой), фильтры на основе FFT (БПФ) и вейвлет-преобразований. Примеры моделирования фильтров рассматриваются ниже.

14.2. Примеры моделирования систем на основе пакета SPB

14.2.1. Модель адаптивного фильтра RLS

Цифровые средства открывают путь к широкому применению адаптивных фильтров, свойства которых подстраиваются под особенности входного сигнала, особенно если они априорно известны. Рисунок 14.30 иллюстрирует технику адаптивной фильтрации для смеси синусоидального сигнала с шумом.

Представленные на рис. 14.30 характеристики фильтра даны в виде стоп-кадра. На самом деле можно видеть, как они меняются во времени, подстраиваясь под фильтрацию сигнала. Осциллограммы дают представление о временных зависимостях сигнала и о погрешности, вызванной шумами. Фильтрация основана на быстром преобразовании Фурье и фактически сводится к подстройке граничной частоты фильтра под частоту поданного сигнала.

14.2.2. Модель адаптивного фильтра Калмана

Еще один пример построения модели адаптивного фильтра представлен на рис. 14.31. На этот раз используется фильтр Калмана, позволяющий осуществлять очистку нестационарного сигнала от шума.

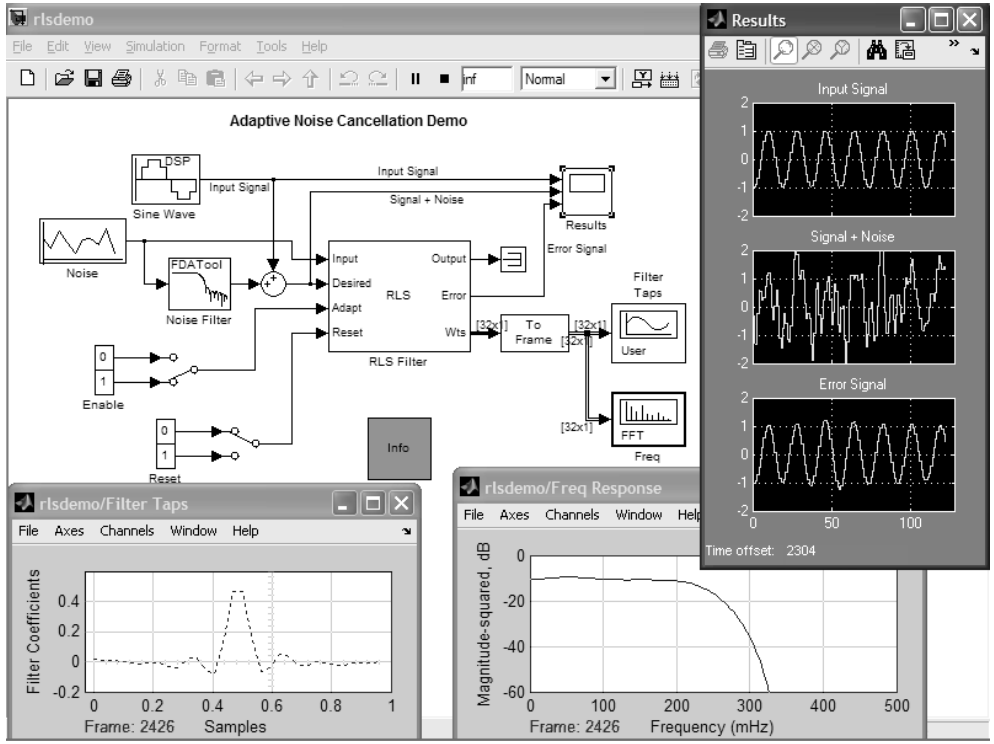


Рис. 14.30. Моделирование адаптивного RLS-фильтра

14.2.3. Модель стереоэкспандера

Иногда возникает необходимость в повышении качества звучания монофонических сигналов. Напомним, что до сих пор монофоническое звучание используется в телевидении и в радиовещании в диапазонах длинных, средних и коротких волн. Простая модель стереоэкспандера, превращающего одноканальное монофоническое звучание в двухканальное псевдостереофоническое звучание, представлена на рис. 14.32.

В основу экспандера положен переход от одноканального тракта в двухканальный с использованием во втором тракте блока линии задержки. С помощью блоков Play Input и Play Output можно прослушать фрагмент музыкального произведения в оригинале и в случае псевдостереофонического звучания.

14.2.4. Модель анализатора спектра с оконным БПФ

Обычный спектральный анализ основан на вычислении зависимостей магнитуды или спектральной плотности сигналов в функции от частоты. Однако он непригоден для нестационарных сигналов и не способен по спектру определять особенно-

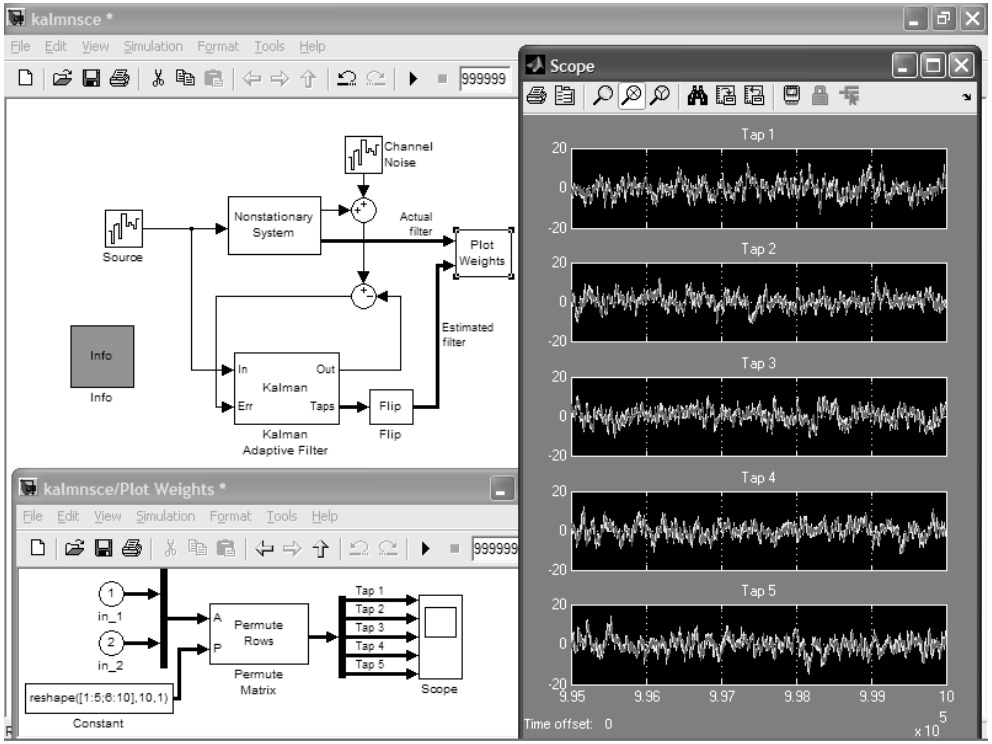


Рис. 14.31. Моделирование адаптивного RLS-фильтра

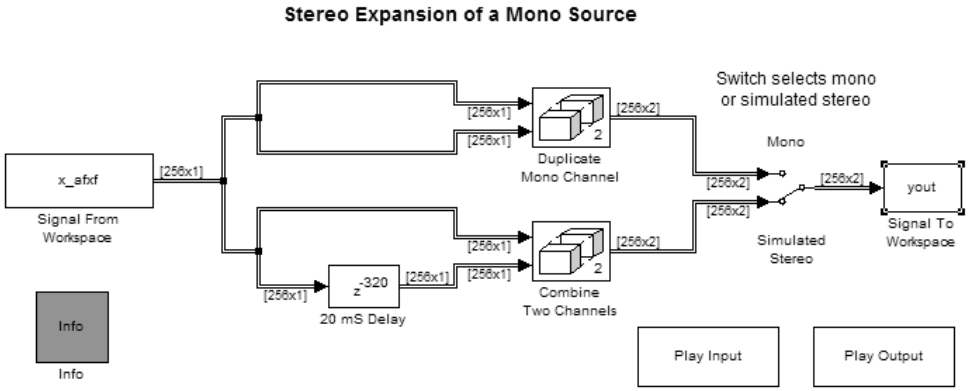


Рис. 14.32. Моделирование стереоэкспандера

сти сигналов в различные моменты времени. Этот недостаток был устранен с разработкой оконного быстрого преобразования Фурье (БПФ). При нем сигнал анализируется методом БПФ в сравнительно узком окне, которое дискретно пробегает по всему временному отрезку действия сигнала. Это позволяет строить спектрограмму в координатах магнитуда – частота – время. Магнитуда при этом отображается цветом или интенсивностью серого цвета.

Модель анализатора спектра, реализующего оконное БПФ, представлена на рис. 14.33. Первая часть модели реализует обычный спектральный анализ методом БПФ и выводит (под ней) обычный спектр – зависимость магнитуды от частоты. Вторая часть реализует оконное БПФ и выводит спектрограмму БПФ. Она показана справа и внизу.

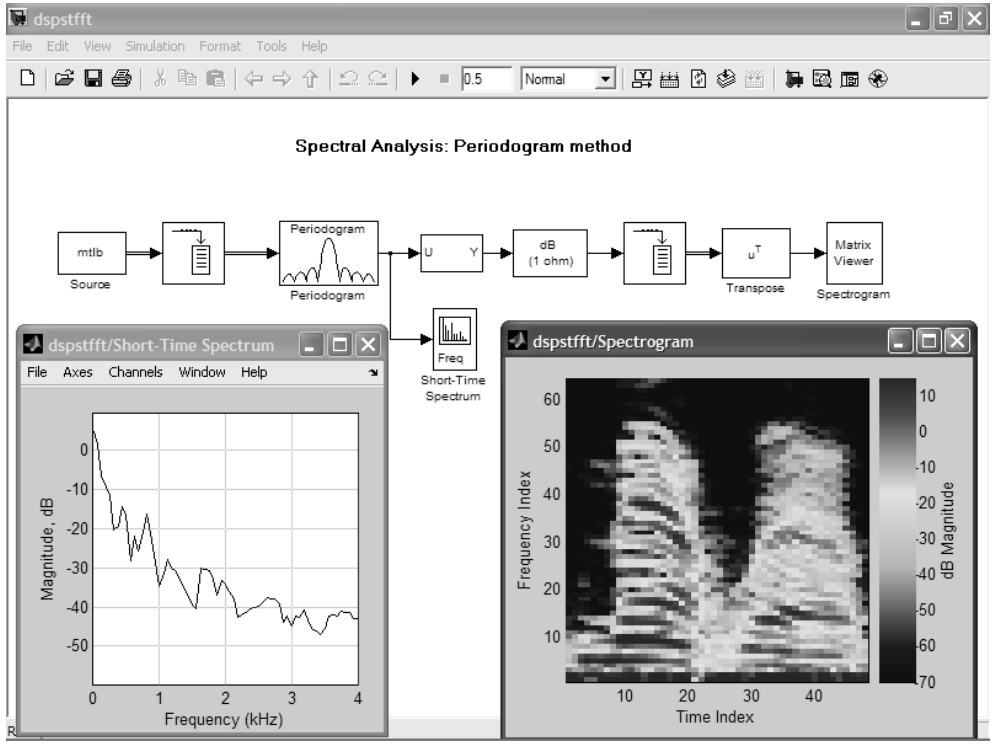


Рис. 14.33. Моделирование анализатора спектра с обычным и оконным БПФ

Следует отметить, что обычные анализаторы спектра реализуют обычное и оконное БПФ, но редко способны строить спектрограмму. На это способны очень дорогие и редкие приборы, например анализаторы спектра реального времени корпорации Tektronix серии 6000. Однако эти приборы предназначены для анализа спектров СВЧ сигналов (до 18 ГГц) и стоят порядка $100 000 и более.

14.2.5. Реконструкция сигнала после вейвлет-фильтрации

К числу перспективных направлений обработки сигналов в наше время относятся так называемые вейвлет-преобразования. Теория и практика вейвлет-преобразований, в том числе с применением пакета расширений системы MATLAB по вейвлетам, детально описаны в книге автора [41]. Поэтому ниже мы ограничимся парой примеров применения технологии вейвлетов.

Обычно фильтрация с помощью вейвлетов сводится к ограничению числа их коэффициентов. Чисто формально это подобно ограничению числа гармоник при фильтрации сигналов с помощью ограниченного ряда Фурье. Однако при этом форма отфильтрованного сигнала не получается в виде плавной кривой. На рис. 14.34 показана простая модель реконструкции сигнала в виде плавной кривой после вейвлет-фильтрации. Результат последней записан в рабочей области MATLAB, извлекается из нее блоком hir и отображается виртуальным осциллографом Scope1.

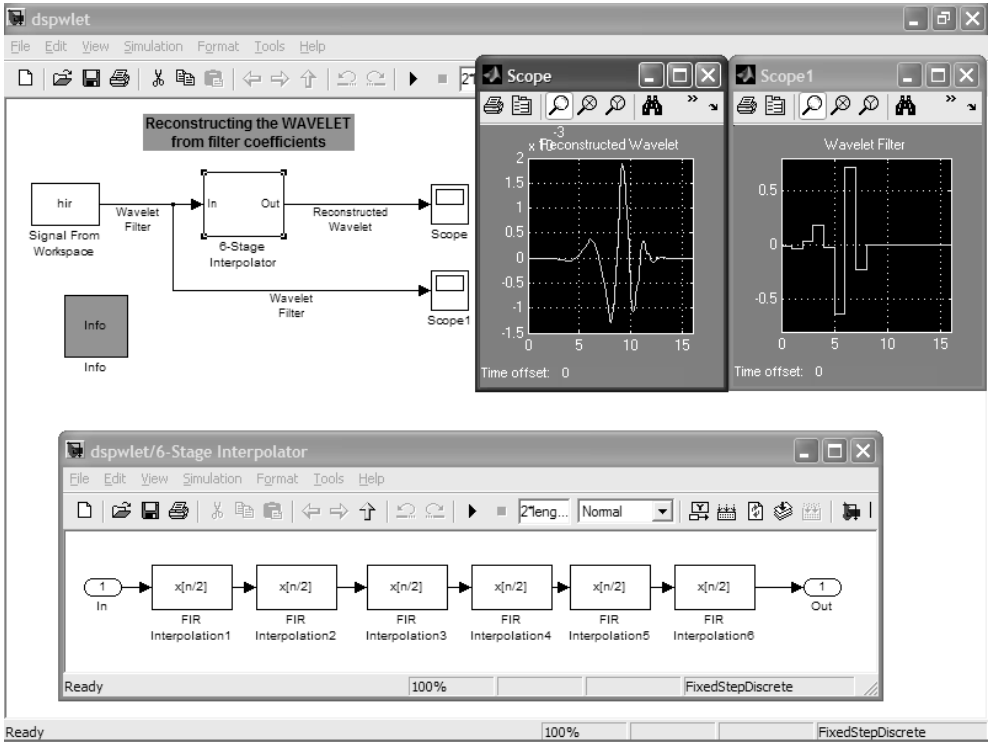


Рис. 14.34. Модель реконструкции сигнала после вейвлет-фильтрации

Реконструкция осуществляется с помощью шестистадийного интерполятора, субблок которого представлен на рис. 14.33 под диаграммой модели. Результат реконструкции отображает виртуальный осциллограф Scope. Реконструкция основана на применении цифровых фильтров типа FIR.

14.2.6. Реконструкция сигнала после вейвлет-фильтрации

Другой пример (рис. 14.35) демонстрирует удивительную (но строго теоретически обоснованную) способность практически полного восстановления сложного сигнала после прямого и обратного многоуровневого вейвлет-преобразования. Здесь, в отличие от предыдущего примера, noicedopp считывает из рабочего пространства сложный сигнал – зашумленный сигнал эффекта Доплера. С помощью блока трехуровневого вейвлет-фильтра осуществляется прямое вейвлет-преобразование, а с помощью блока реконструкции – обратное вейвлет-преобразование.

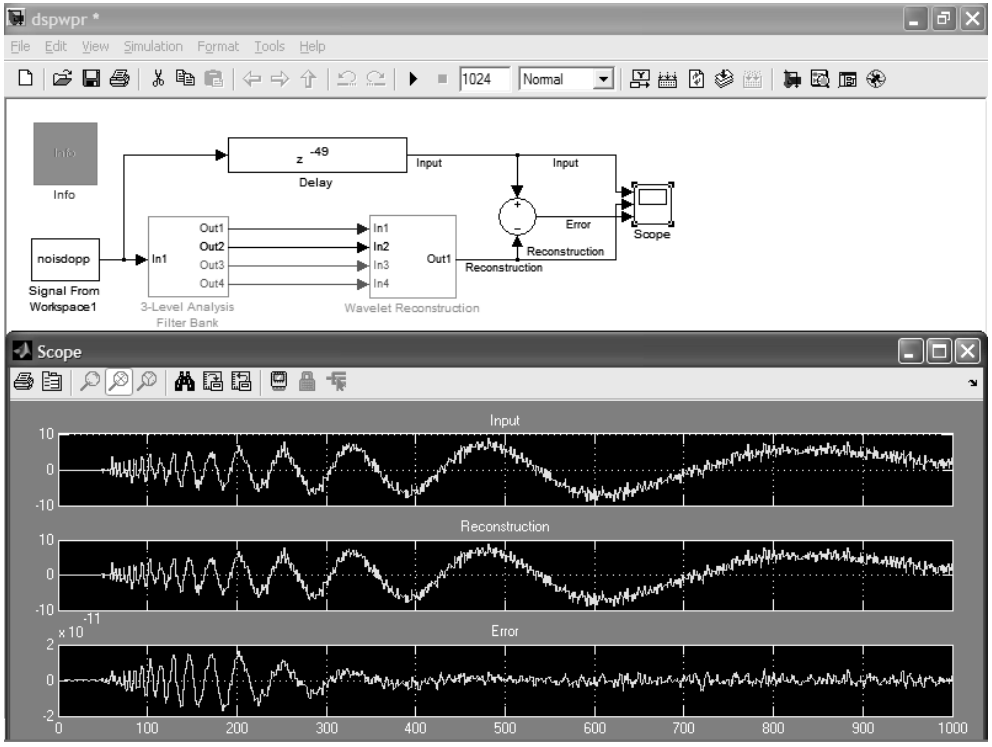


Рис. 14.35. Модель реконструкции сигнала после прямого и обратного многоуровневых вейвлет-преобразований

Трехлучевой виртуальный осциллограф выводит осциллограммы (они показаны снизу) исходного сигнала, сигнала после вейвлет-преобразований и разностного сигнала (сигнала ошибки). Первые две осциллограммы на глаз не выделяют ошибки. Малая погрешность реконструированного сигнала ясна из третьей осциллограммы – видно, что она по модулю меньше $2 \cdot 10^{-11}$, то есть ничтожно мала.

14.2.7. Вейвлет-очистка сигнала от шума

Уже давно большое значение имеет очистка от шума нестационарных сигналов, например звуковых. Заметим, что нестационарными являются сигналы с изменяемыми во времени параметрами, например амплитудой, частотой или фазой. В этом случае непригодны методы, основанные на использовании свойств стационарных сигналов. Одними из новых и перспективных являются методы, основанные на применении вейвлет-преобразований.

На рис. 14.36 показана диаграмма модели очистки от шума сигнала с изменяющейся во времени частотой и амплитудой. Очистка основана на ограничении числа уровней вейвлет-преобразований в зависимости от уровня вейвлет-коэффициентов. Такой вариант очистки обеспечивает меньшие искажения сигналов.

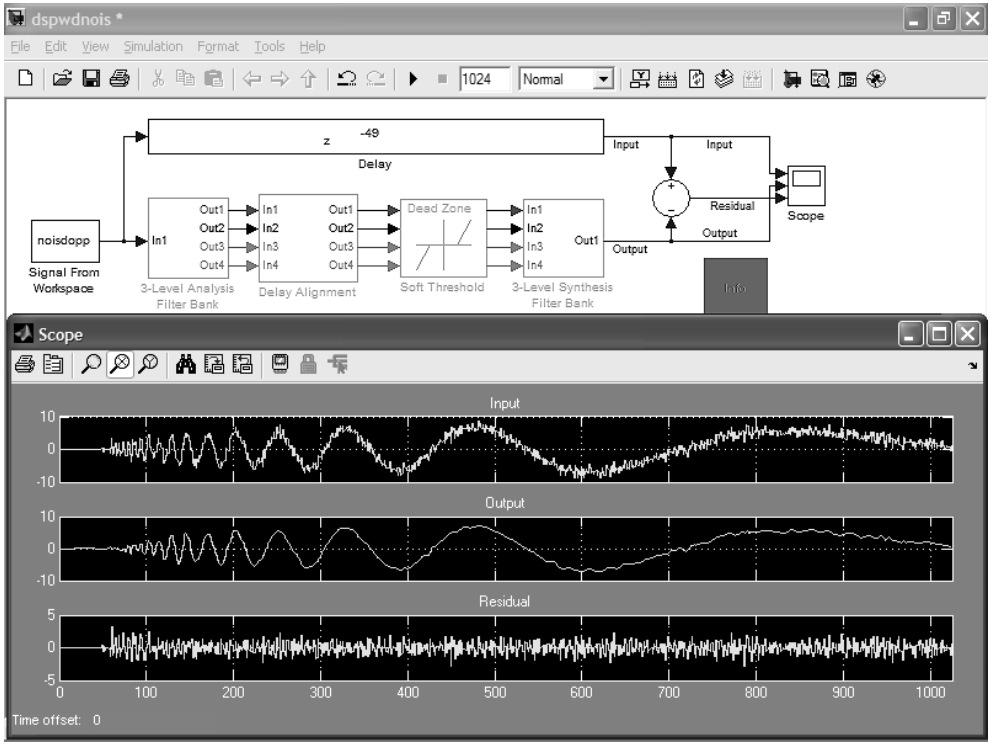


Рис. 14.36. Модель вевлет-очистки сигнала от шума

Следует предостеречь читателя от эйфории по поводу этого нового метода очистки сигналов от шума. Модель вейвлет-очистки только кажется простой. На практике ее реализация требует прекрасного знания довольно сложных математических понятий, относящихся к вейвлетам и реализации вейвлет-преобразований в виде далеко не простых устройств. Впрочем, в наше время уже выпускаются интегральные микросхемы для вейвлет-преобразований, а сами они положены в основу нового стандарта сжатия аудиосигналов.

14.2.8. 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 SP и Simulink имеет большое значение. Рисунок 14.37 демонстрирует простейшую модель системы SSB – **Simple Basic Version**. В ней используются источники синусоидальных сигналов пакета **SP** и блоки математических операций, выделяющие боковые полосы (балансный метод SSB).

Результаты моделирования представлены спектрами боковых полос и осциллограммами сигналов этих полос.

Следует отметить, что в этом примере реализована возможность наблюдения за процессом однополосной модуляции в динамике. Представленные результаты получены как стоп-кадр анимационной картины модуляции при ограниченном времени моделирования.

14.2.9. Адаптивная дельта-импульсная кодовая модуляция

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. 14.38 дан пример устройства, которое осуществляет адаптивную дельта-импульсную кодовую модуляцию (ADPCM), заметно превосходящую LDM по качеству передачи быстро изменяющихся сигналов. На рисунке показана основная модель тракта модуляции и подсистемы адаптивного модулятора и демодулятора, построенная на основе блоков пакета **SP**.

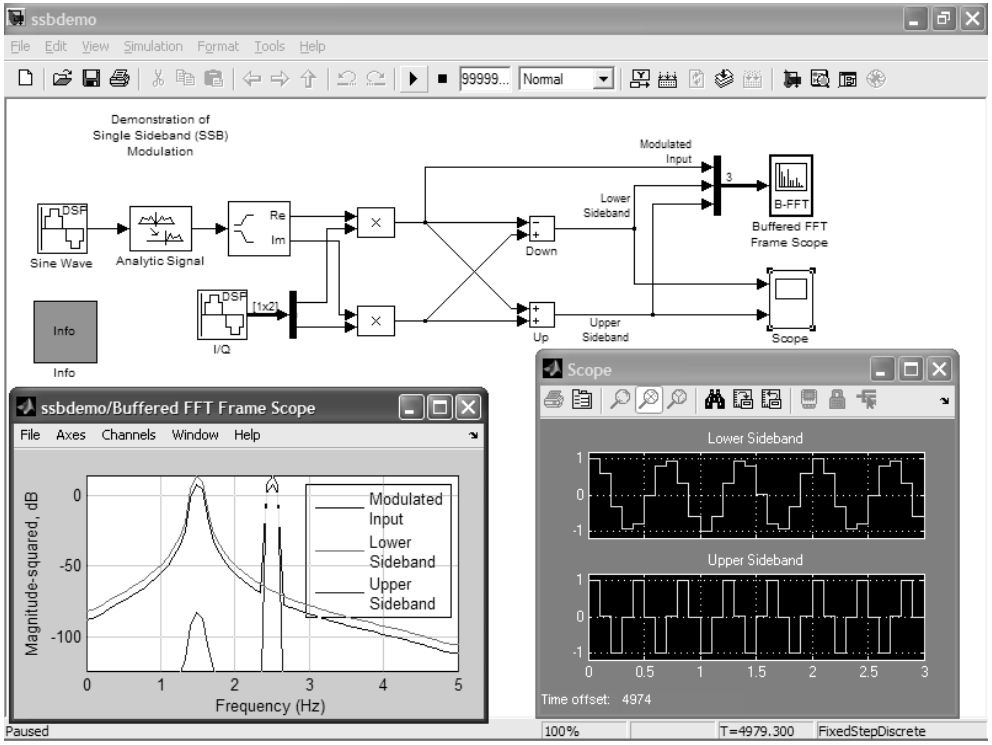


*Рис. 14.37. Модель однополосной модуляции (SSB) – **Simple Basic Version***

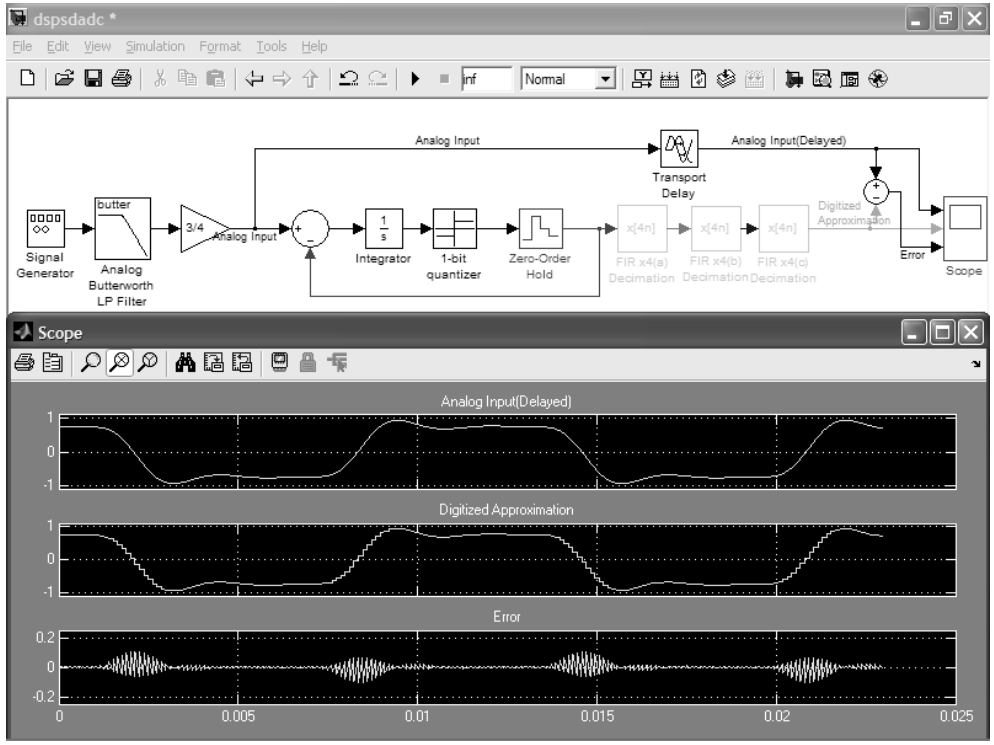


Рис. 14.38. Пример моделирования тракта с ADPCM

На рис. 14.38 показаны также результаты моделирования в виде осциллограмм. Верхняя осциллограмма показывает исходный сигнал и наложенный на него сигнал, прошедший тракт модуляции-демодуляции.

14.3. Пакет расширения RF Blockset

14.3.1. Назначение пакетов расширения RF Toolbox и Blockset

Пакеты расширения по обработке сигналов и анализу и проектированию коммуникационных устройств не приспособлены для проектирования радиочастотных (Radio Frequence или RF) цепей, особенно в СВЧ-диапазоне и там, где применяются цепи и устройства с сосредоточенными постоянными. Этот недостаток был устранен в новых реализациях MATLAB введением специальных пакетов расширения RF Toolbox и Blockset. Они могут использоваться как совместно, так и отдельно и служат для решения следующих задач:

- чтение и запись данных радиочастотных устройств в различных форматах (.snp, .ynp, .znp, .hnp и .AMP-формате фирмы MathWorks);
- преобразование систем S, Y, Z, h, T и ABCD-параметров схем;
- построение данных в плоскости X–Y, в полярных координах и в виде диаграмм Смита;
- вычисление и применение S-параметров цепей;
- вычисление коэффициентов отражения на входе и выходе цепей и положения волн отражения (VSWR);
- специальные виды цепей, например каскадные, гибридные, параллельные и последовательные.

При этом используются указанные ниже модели:

- пассивные цепи и основные элементы цепей, описываемые файлами с расширениями .snp, .ynp, .znp и .hnp;
- последовательные и параллельные RLC-цепи;
- усилители и микшеры, описываемые файлами с расширениями .s2p, .y2p, .z2p, .h2p и .amp;
- линии передачи, модели которых основаны на их геометрических характеристиках (например, коаксиальные и полосковые линии передачи);

- LC-лестничные фильтры различного типа (НЧ, ВЧ, полосовые пропускающие и заграждающие и т. д.), основанные на интерактивном представлении их электрических параметров.

С этими моделями и компонентами можно создавать различные типы радиочастотных цепей, например каскадные, гибридные, параллельные и последовательные цепи, и выполнять с ними такие операции, как:

- анализ параметров сетей и определение их частотных параметров;
- вычисление нужных иных параметров;
- вывод листинга параметров;
- графическая визуализация параметров и работы цепей;
- импорт и экспорт параметров;
- выполнение интерактивного анализа и проектирования радиочастотных цепей.

14.3.2. Системы параметров для радиочастотных объектов

В пакете **RF Toolbox** используются хорошо известные матричные методы анализа радиочастотных цепей и устройств. В частности, широко используются методы, основанные на представлении цепей в виде четырехполюсников, имеющих входы и выходы. Интерес представляют как устройства, представляемые одним четырехполюсником (например, усилители или неделимые фрагменты линий передачи), так и каскадные схемы, построенные последовательным соединением отдельных каскадов (многокаскадные усилители, лестничные многозвенные фильтры и т. д.).

В радиотехнической литературе широко используются системы Y-, Z-, H- и ABCD-параметров четырехполюсников [18]:

$I_1 = Y_{11}E_1 + Y_{12}E_2 \quad I_2 = Y_{21}E_1 + Y_{22}E_2$ – система Y-параметров;

$E_1 = Z_{11}I_1 + Z_{12}I_2 \quad E_2 = Z_{21}I_1 + Z_{22}I_2$ – система Z-параметров;

$E_1 = H_{11}I_1 + H_{12}E_2 \quad I_2 = H_{21}I_1 + H_{22}E_2$ – система H-параметров;

$$\begin{bmatrix} E_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E_2 \\ I_2 \end{bmatrix}$$ – система ABCD-параметров.

Радиочастотные цепи могут быть описаны в этих системах параметров, уравнения для которых связывают входные ток I_1 и напряжение E_1 с выходными током I_2 и напряжением E_2. В указанной литературе описаны способы вычисления параметров Y, Z, H и ABCD и формулы преобразования этих параметров. Они и положены в основу встроенных в пакет **RF Toolbox** функций.

Определение параметров в указанных системах параметров четырехполюсников основано на использовании опытов короткого замыкания и холостого хода. Однако в СВЧ-цепях и устройствах осуществление этих опытов практически неосуществимо. Кроме того, в этой области обычно используется соединение четырехполюсников с генератором входного сигнала и нагрузкой, а также между собой с помощью линий передачи с дискретными и распределенными постоянными.

При этом волновое сопротивление линий обычно бывает стандартным и равным 50, 75 или 100 Ом. В этих условиях применение представленных параметров оказывается неудобным, а подчас и просто невозможным. Особенно это касается цепей с каскадным соединением компонентов с тремя и более полюсами.

Для СВЧ-цепей обычно используется система S-параметров, представляющих матрицы рассеивания. Матрицы рассеивания определяют взаимосвязь между переменными a_n (пропорциональными амплитудам входящих волн на n-ой паре полюсов) и переменными b_n (пропорциональными амплитуде выходящих волн n-ой пары полюсов). Для четырехполюсника:

$b_1 = S_{11}a_1 + S_{12}a_2 \quad b_2 = S_{21}a_1 + S_{22}a_2$ – система S-параметров.

Или в общем случае **b**=**Sa**.

Эта система параметров недостаточно удобна при вычислении параметров каскадных устройств. Поэтому используется еще одна система T-параметров, или параметров передачи:

$$\begin{bmatrix} b_1 \\ a_1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}$$ – система T-параметров.

Система S-параметров в пакете **RF Toolbox** принята как основная (по умолчанию), но могут использоваться и другие системы параметров. Токи, напряжения и параметры в этих системах рассматриваются как комплексные величины.

14.3.3. Библиотека блоков пакета RF Blockset

Основой пакета RF Blockset является библиотека его блоков. Она вызывается из командной строки MATLAB подачей команды:

```
>> rflib
```

Это приводит к появлению основного окна библиотеки, представленного на рис. 14.39. Нетрудно заметить, что в нем имеются три набора блоков:

- `Mathematical` – набор математических блоков;
- `Physical` – набор физических блоков;
- `Demos` – переход в окно **Demos** справки по пакету **RF Blockset**.

На рис. 14.40 показаны окна разделов библиотеки пакета расширения **RF Blockset** по фильтрам, линиям передачи и портам ввода/вывода, на рис. 14.41 – окна разделов библиотеки с блоками усилителей и миксеров.

Наиболее полной является раздел библиотеки физических устройств пакета расширения **RF Blockset**. В этом окне представлены следующие наборы блоков физических устройств:

- `Ladder Filters` – блоки ступенчатых фильтров (рис. 14.40 сверху);
- `Transmission Lines` – блоки линий передачи (рис. 14.40 снизу);
- `Black Box Elements` – блоки «черных ящиков» (рис. 14.39 слева внизу);
- `Amplifiers` – усилители (рис. 14.41 слева);

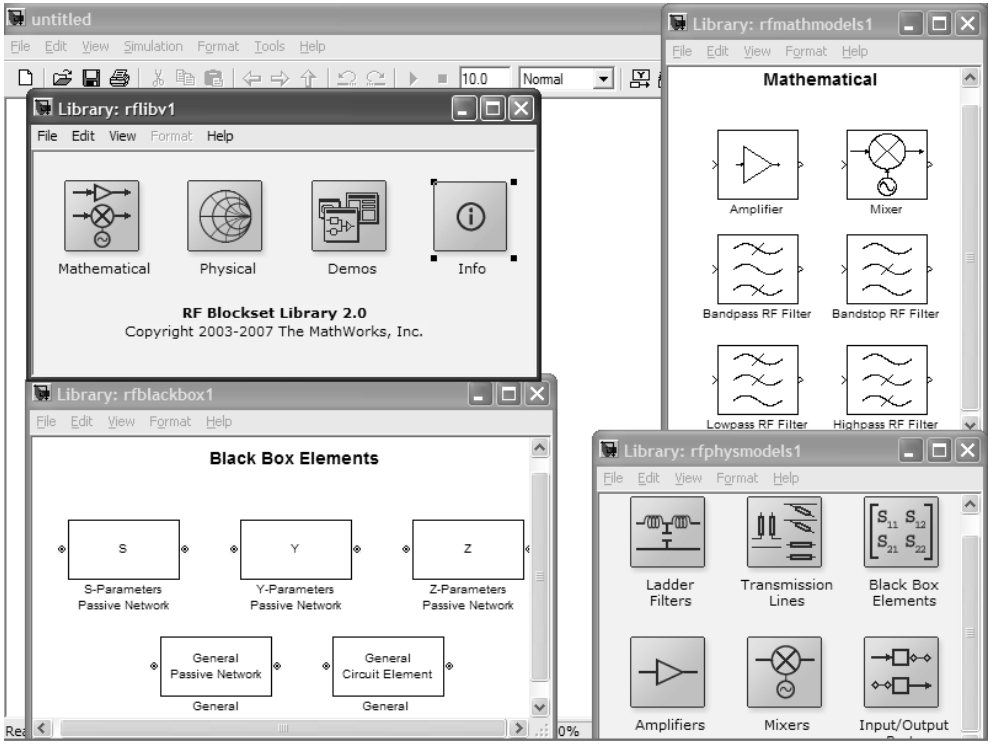


*Рис. 14.39. Основное окно библиотеки пакета расширения **RF Blockset** и разделы библиотеки по математическим и физическим блокам*

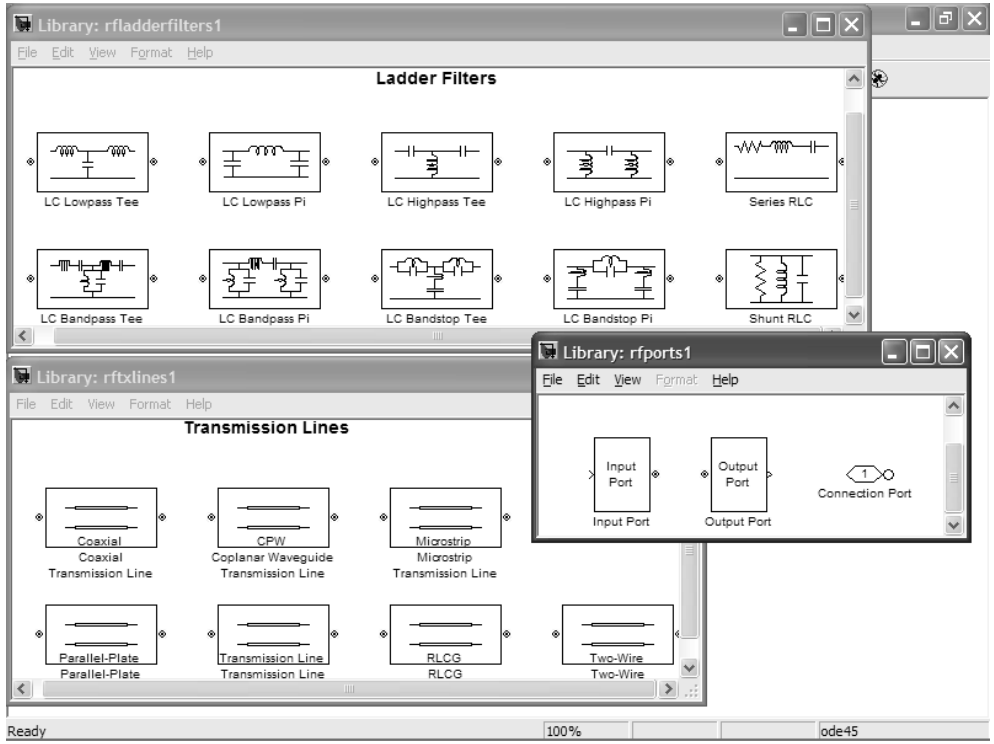


*Рис. 14.40. Окна разделов библиотеки пакета расширения **RF Blockset** по фильтрам, линиям передачи и портам ввода/вывода*

- `Mixers` – миксеры (рис. 14.41 справа);
- `Input/Output Ports` – порты ввода/вывода (рис. 14.40 – малое окно).

В целом можно отметить, что библиотека данного пакета содержит небольшое число блоков, но надо принять во внимание, что это именно те блоки, которые учитывают специфику радиочастотных цепей.

14.3.4. Работа с математическими блоками

Набор математических блоков пакета **RF Blockset** сравнительно невелик – он насчитывает всего шесть блоков, представленных на рис. 14.39 в окне, показанном справа и сверху.

Особое значение принадлежит блоку «математического» усилителя Amplifier. Он имеет ряд характерных параметров. Параметр метода со значением Linear задает линейный усилитель. Но из списка можно выбрать виды нелинейности для построения нелинейного усилителя – это кубический полином, гиперболический тангенс и еще три специальных вида нелинейности. Параметр Linear Gain задает (в децибелах, по умолчанию 0) коэффициент передачи линейного усилителя. Па-

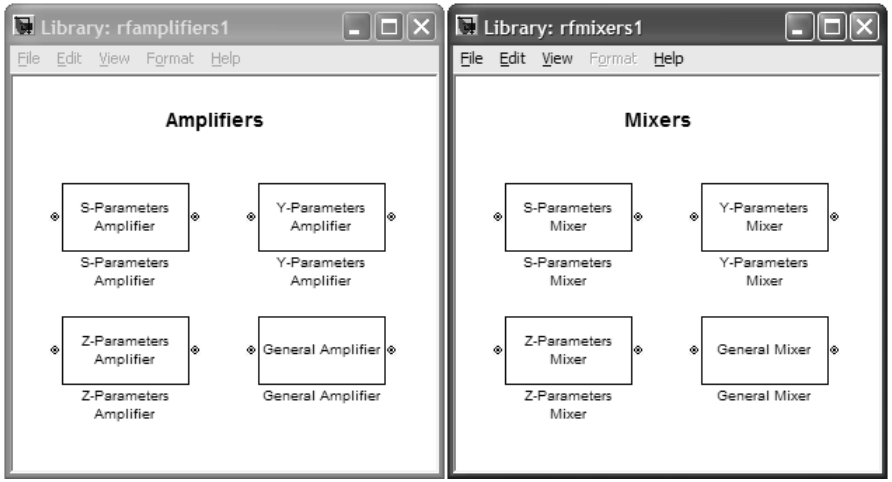


*Рис. 14.41. Окна разделов библиотеки пакета расширения **RF Blockset** по усилителям и миксерам*

раметры Specification method и Noise factor задают вид шума и его уровень, что обеспечивает моделирование шумящего усилителя. Возможно задание фактора шума (по умолчанию), шумовой температуры или шума, выраженного в децибелах. Возможна также инициализация числом (по умолчанию 67 987) генератора шума.

Простейший пример применения усилителя для усиления шумового сигнала от блока генератора белого шума показан на рис. 14.42. Для контроля входного сигнала (а это шум) используется анализатор спектра. Шумовая дорожка на выходе усилителя колеблется около уровня 0 дБ, что указывает на единичное усиление усилителя по умолчанию. Поскольку белый шум имеет одинаковый средний уровень шума на всех частотах, то шумовая дорожка располагается горизонтально. Это говорит об отсутствии амплитудно-частотных искажений у модели усилителя.

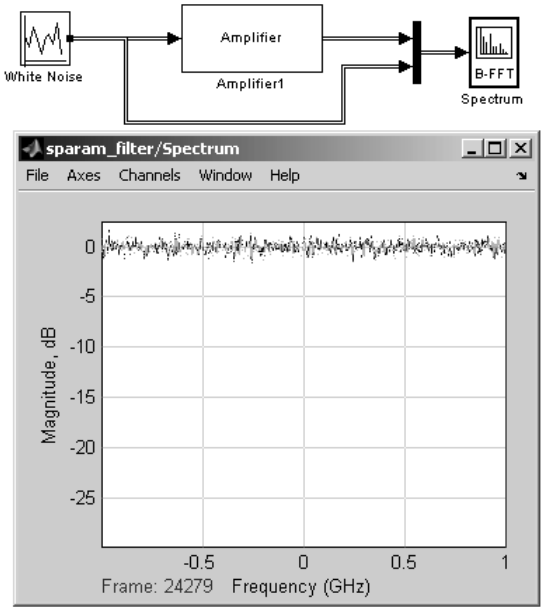


Рис. 14.42. Простейшая радиочастотная система, содержащая генератор белого шума, усилитель и анализатор спектра

Блок **Mixer** имеет окно со следующими параметрами: уровень ослабления, уровень фазового шума, частота отсечки и число, инициирующее генератор случайных чисел в источнике шума.

Кроме того, в данном наборе блоков представлены четыре типовых фильтра: НЧ, ВЧ, полосовой пропускающий и полосовой задерживающий. Обратите внимание на их наглядное обозначение, в котором фигурируют три синусоиды. Перечеркнуты те из них, которые не пропускаются фильтром. Параметры фильтров

достаточно полно описывались, поэтому ограничимся примером построения двух вариантов полосовых пропускающих фильтров, представленных на рис. 14.43 и имеющих порядок, равный 3 и 5.

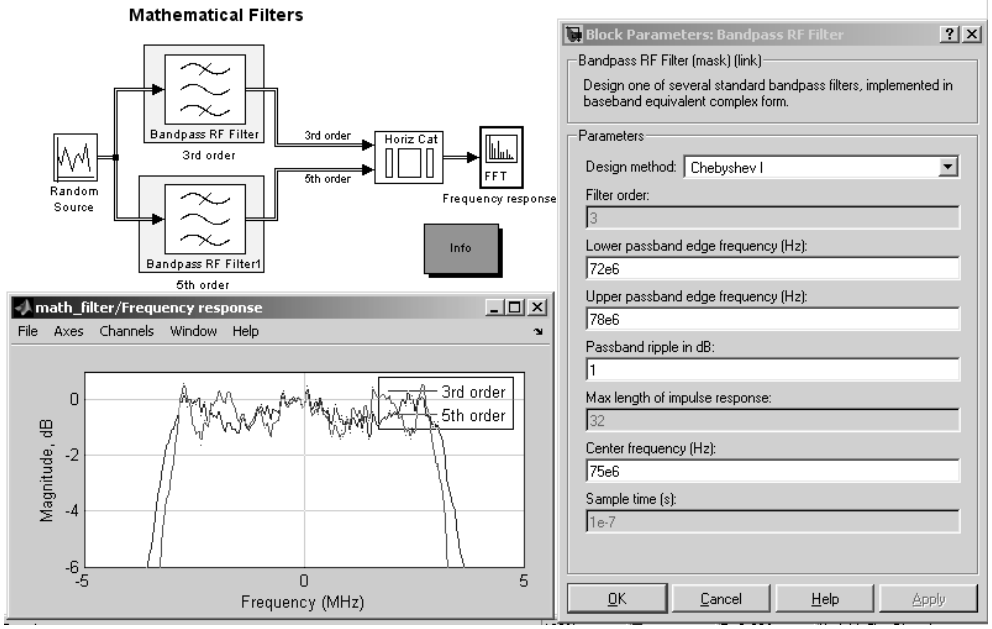


Рис. 14.43. Модели двух математических полосовых пропускающих фильтров порядка 3 и 5

На рис. 14.43 справа показано окно установки параметров данного фильтра. А под самой моделью приведены спектрограммы шумового сигнала на выходе фильтров. Поскольку белый шум имеет равномерный спектр, то спектр на выходе любого фильтра с неизбежными шумовыми пульсациями представляет АЧХ фильтра. Такой метод построения АЧХ широко используется в СВЧ-технике, где создание весьма широкополосных генераторов СВЧ-сигналов очень сложно, а подчас и невозможно. В то же время генератор белого шума на практике реализуется достаточно просто. Именно поэтому во всех примерах применения пакета **RF Blockset** в качестве тестовых сигналов используются генераторы шума.

14.3.5. Применение блоков портов ввода/вывода

Если внимательно присмотреться к входам и выходам блоков, то можно заметить, что они имеют разное обозначение. Математические блоки имеют входы и выходы, помеченные маленьким знаком >. Такое же обозначение имеют блоки пакетов

расширения Simulink, DSP и др. Это указывает на принципиальную возможность их соединения друг с другом. Реальная возможность существует не всегда из-за обилия форматов передаваемых данных.

Однако блоки физических устройств пакета **RF Bkockset** имеют иное обозначение – в виде точки внутри окружности. Это обозначение напоминает разрез коаксиальной линии передачи и указывает на особый тип входов и выходов. Структура передаваемых данных показана на рис. 14.44 на примере усилителя, описанного S-параметрами. Напрямую соединять такие блоки с обычными нельзя, и редактор модели просто отказывается это делать.

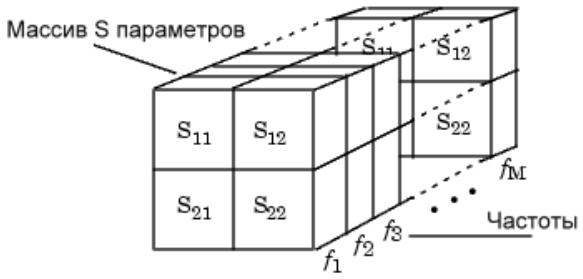


Рис. 14.44. Структура данных усилителя, описанного S-параметрами

В связи с этим в состав блоков пакета **RF Bkockset** введены особые согласующие блоки. Блок **Input Port** служит для подключения выхода обычного блока к входу блока физических устройств пакета **RF Bkockset**. А блок **Output Port** – для подключения выхода блока физических устройств пакета **RF Bkockset** ко входу обычного блока. Рисунок 14.45 иллюстрирует применение этих несложных правил.

14.3.6. Визуализация графических характеристик блоков

Некоторые блоки, например усилители, характеризуются множеством графических характеристик, например частотными характеристиками S-параметров, передаточными характеристиками и т. д. Для их построения на практике блок подключается к генератору испытательных сигналов и регистратору, например вольтметру переменного напряжения. Испытания приходится проводить в широком диапазоне частот, что часто делает их нереальными из-за отсутствия необходимой измерительной аппаратуры. К тому же такая аппаратура для снятия специальных характеристик блоков (например, спектров или диаграмм Смита) очень дорога и есть далеко не во всякой радиотехнической лаборатории.

Казалось бы, данную проблему при математическом моделировании решить легко, и примеры этого для пакетов расширения **Signal Processing Toolbox**,

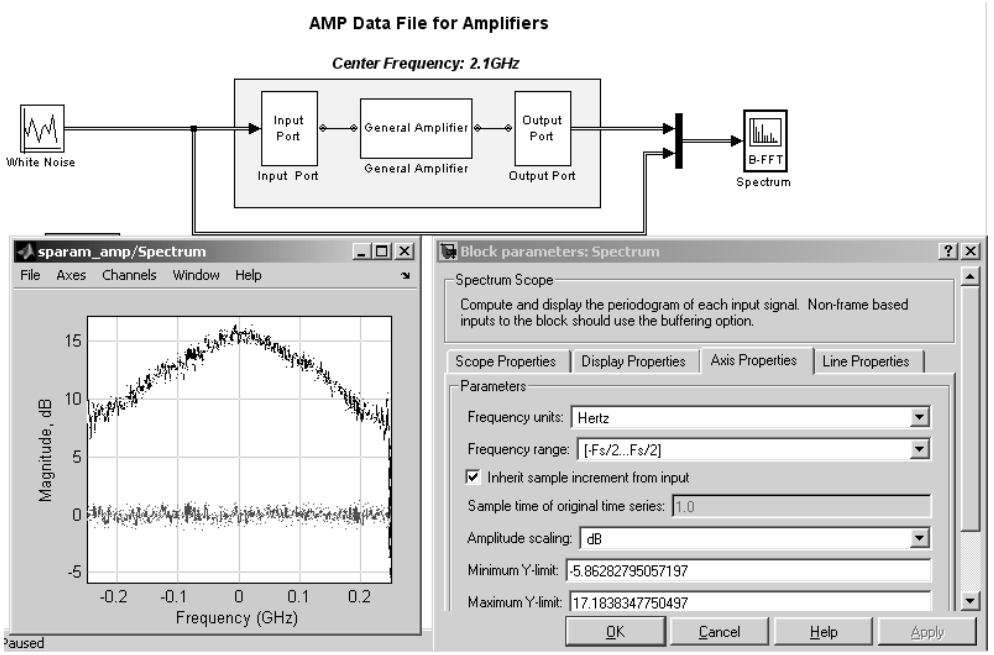


Рис. 14.45. Пример применения блоков входа/выхода

Control System Toolbox и других неоднократно приводились. Однако для блоков физических устройств пакета **RF Toolbox** (даже при использовании блоков ввода/вывода) получить нужные характеристики нелегко из-за больших различий в форматах используемых данных и сигналов.

В пакете **RF Blockset** эта трудность преодолена за счет использования функций пакета **RF Toolbox**. При этом «тайный» доступ к ним реализован прямо с окон задания параметров физических блоков. Никаких программных фрагментов для построения нужных характеристик писать или вводить в командном режиме не требуется.

В справке по пакету **RF Blockset** приводится множество примеров построения графических характеристик физических блоков. Мы ограничимся парой наглядных примеров, что позволит читателю самому выбирать и строить нужные графические зависимости.

Вернемся к модели, представленной на рис. 14.45, и попытаемся построить характеристики блока усилителя **General Amplifier**. Для этого прежде всего надо остановить процесс моделирования, поскольку без этого изменения параметров в окне параметров усилителя будут невозможны. На рис. 14.46 показана рассмотренная модель после остановки моделирования и вывода окна свойств усилителя – оно показано снизу и слева.

Чтобы построить ту или иную характеристику, нужно в окне параметров усилителя установить птичку в прямоугольнике опции **Plot the network parameters**

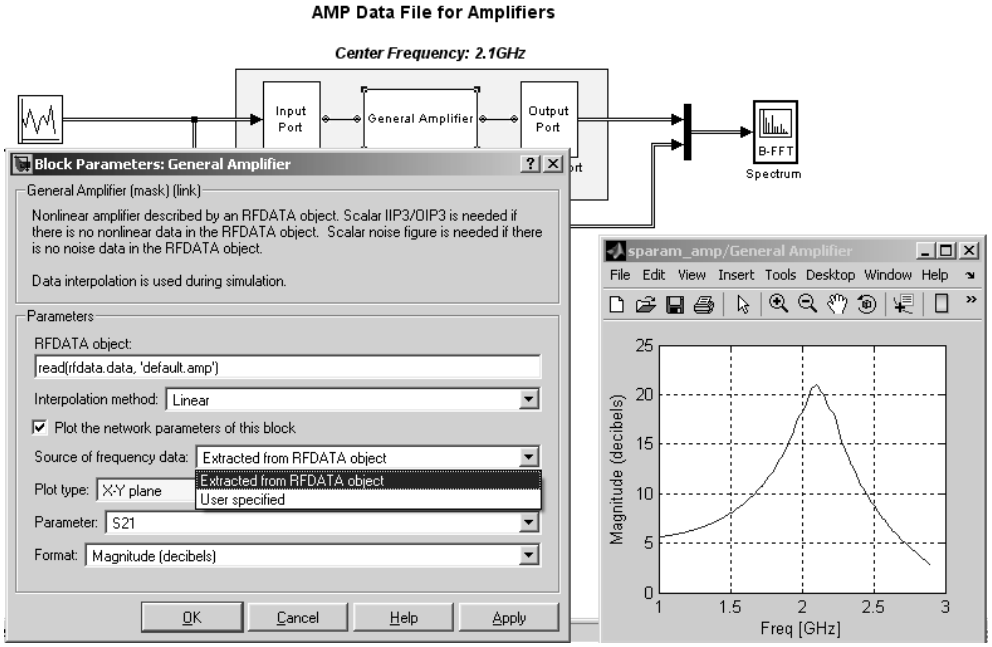


Рис. 14.46. Пример построения частотной зависимости параметра S21 усилителя

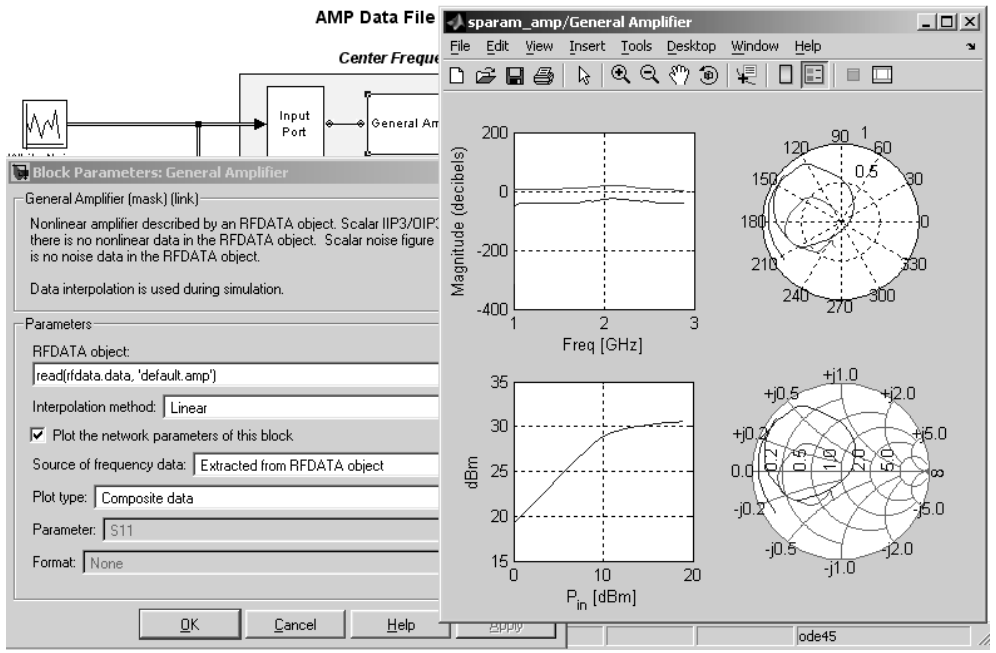


Рис. 14.47. Пример построения нескольких графических характеристик разного вида

of this block (построить график параметра цепи для данного блока). При этом окажутся доступными установки для построения различных характеристик:

- `Sourse of frequence data` – источник для данных частоты (может быть RFData или заданный пользователем);
- `Plot type` – тип графика (например, XY Plot, Polar Plot и т. д.; всего более десятка видов графики, выбираемых из списка);
- `Parameter` – параметр, график которого нужно отобразить;
- `Format` – формат (тип) графика, выбираемый из списка.

На рис. 14.46 представлены установки для построения графика частотной зависимости параметра S21 в прямоугольной системе координат. Сам график построен в нижнем правом углу. Для его построения достаточно нажать кнопку **OK** окна параметров блока **General Amplifier**.

Если надо на одном графике построить более одной характеристики, совместимых по типу графика, то достаточно повторить процедуру построения для других характеристик. Наконец, установив тип графика Composite data, можно построить в одном окне графика сразу несколько графиков разного типа – рис. 14.47.

14.4. Примеры применения пакета RF Blockset

14.4.1. Сравнение реализаций усилителей

Одной из проблем современной радиотехники является искажение сигналов при их прохождении через нелинейные усилители. Линейные усилители не вносят в спектр входного сигнала каких-либо новых спектральных составляющих. Однако таких усилителей на практике просто нет – любой реальный усилитель при больших сигналах становится нелинейным. Представляет интерес моделирование линейных и нелинейных усилителей. Такой пример есть в справке по пакету, и его модель представлена на рис. 14.48.

В данной диаграмме представлены по существу пять моделей усилителей – два математических и два физических усилителя (в терминах пакета **RF Blockset**). На вход усилителей подан двухчастотный сигнал. Результаты моделирования представлены в виде спектрограмм – рис. 14.49. Из левой спектрограммы от-

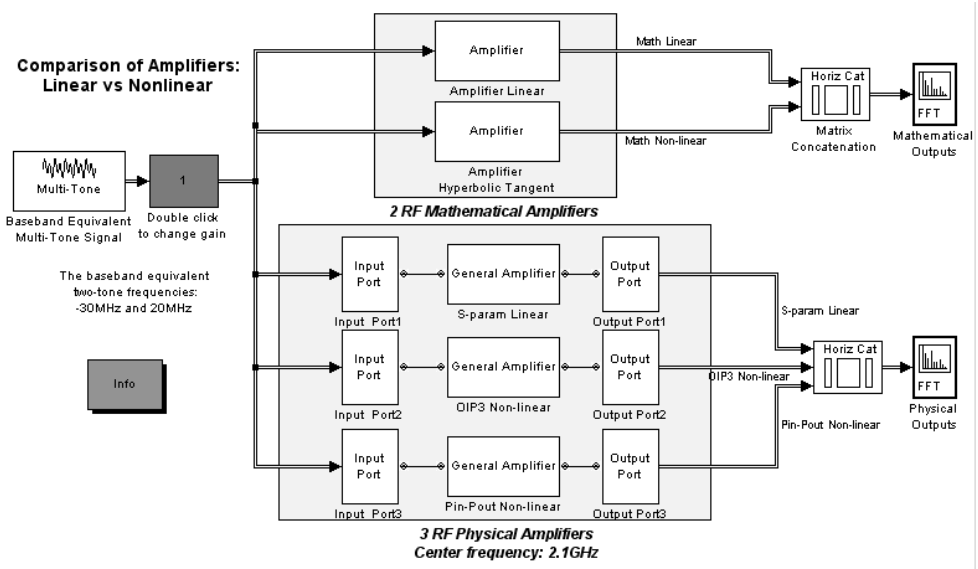


Рис. 14.48. Модель сравнения усилителей

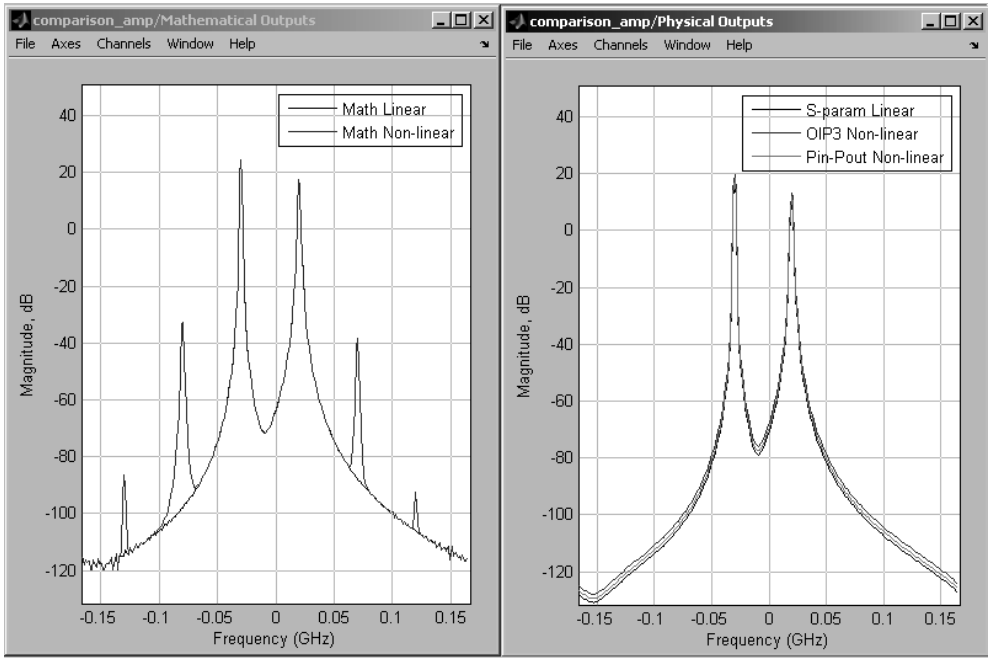


Рис. 14.49. Результаты моделирования прохождения двухчастотного сигнала

четливо видно, что спектр линейных усилителей повторяет спектр входного сигнала, а в спектре выходного сигнала нелинейных усилителей присутствуют сигналы, вызванные интермодуляционными искажениями.

14.4.2. Моделирование фильтров на линиях передачи

В диапазоне СВЧ фильтры часто строятся на основе линий передачи, которые имеют простое конструктивное оформление. На рис. 14.50 представлена диаграмма модели, в которой задано построение двух полосовых фильтров разного порядка – третьего и четвертого. Порядок задает число ячеек линии, которое равно числу элементов векторов, задающих погонные параметры L и C для каждой ячейки линии. На рисунке представлено также окно задания параметров фильтра третьего порядка. Фильтры отличаются неравномерностью амплитудно-частотной характеристики и ослаблением в полосе пропускания.

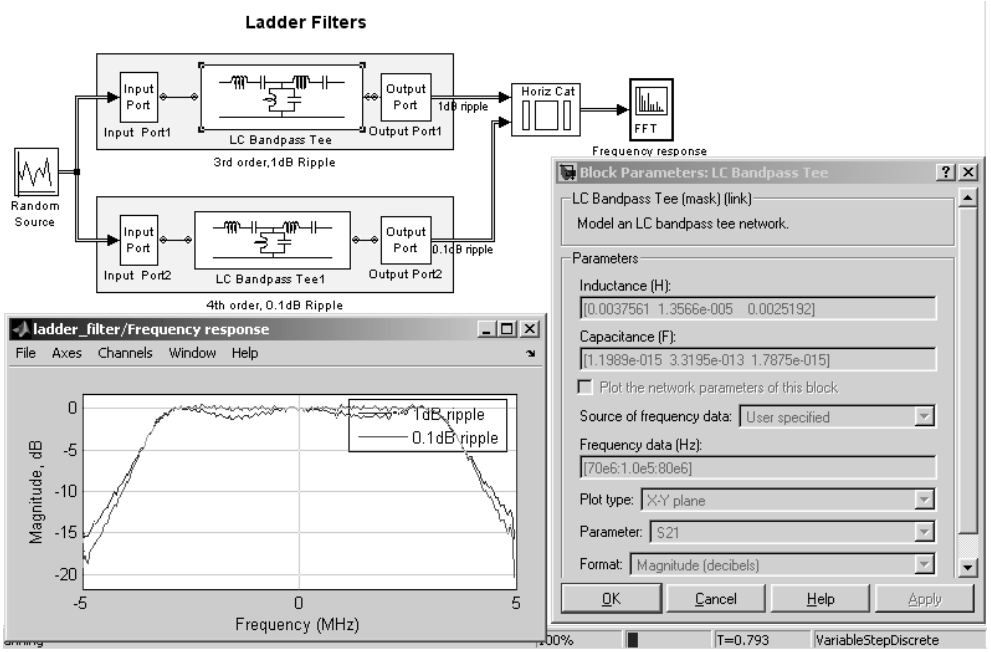


Рис. 14.50. Модель двух полосовых фильтров разного порядка на линиях передачи

На рис. 14.51 представлено окно параметров для фильтра четвертого порядка. В нем включена опция вывода графических зависимостей для параметра S21 – случай вывода композитного графика. Соответствующий график представлен на рис. 34.41 и показывает АЧХ и ФЧХ параметра и их представление в полярной

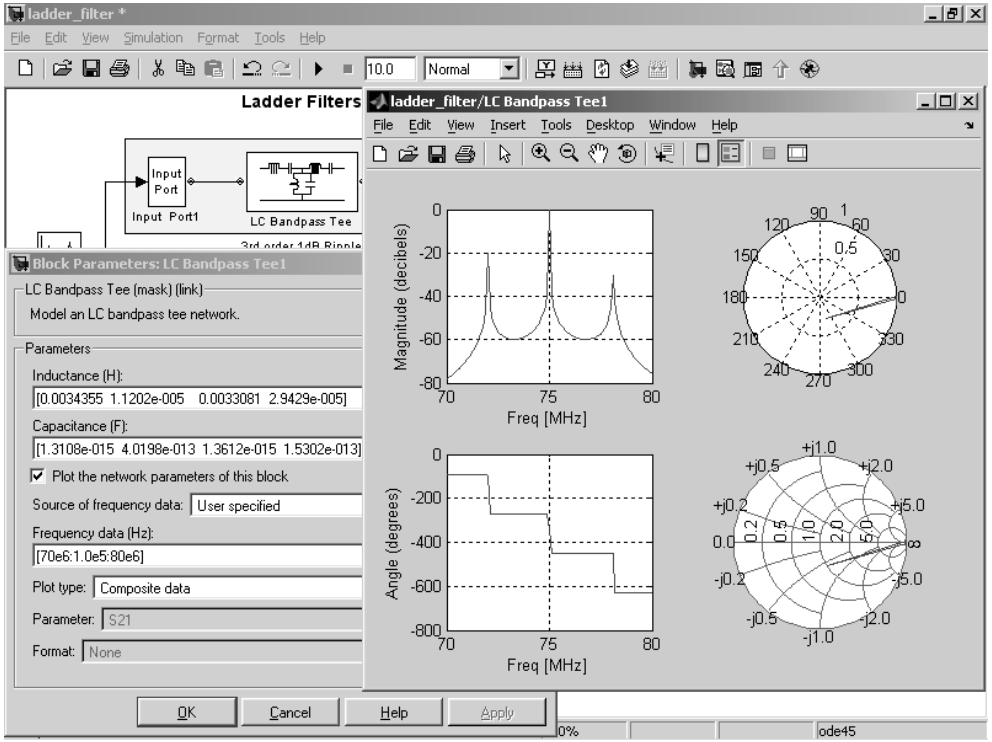


Рис. 14.51. Данные о частотной зависимости параметра S21 полосового фильтра четвертого порядка

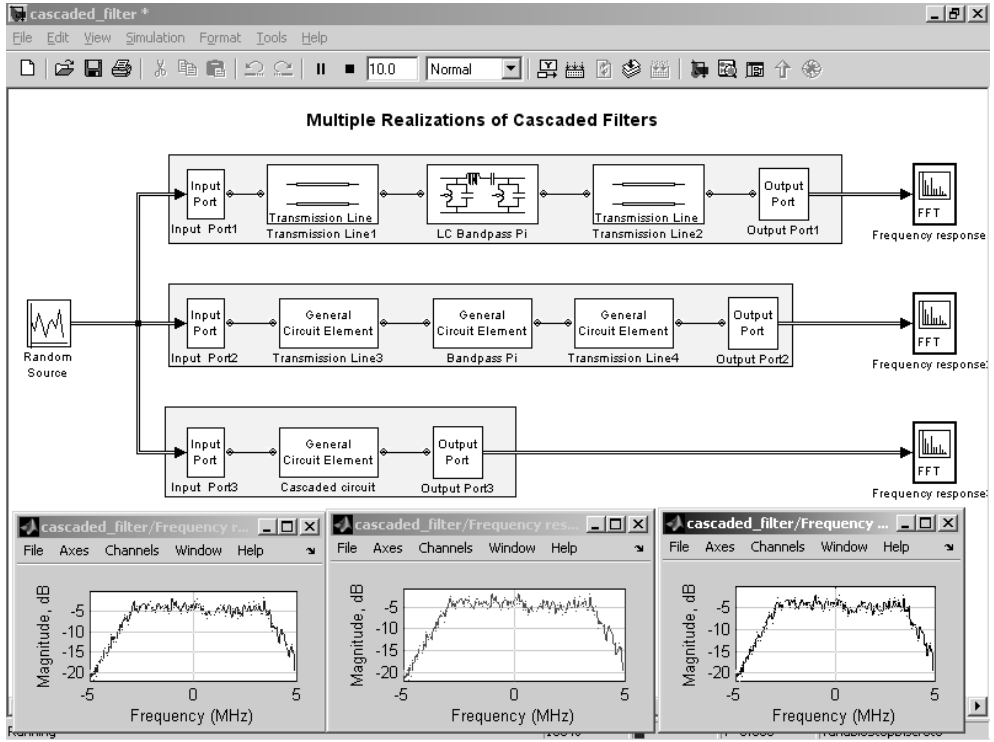


Рис. 14.52. Модель каскадных полосовых фильтров и фильтра на элементе General Circuit Element

системе координат и в виде диаграммы Смита. Это еще раз иллюстрирует эффективность средств анализа радиочастотных устройств.

Рисунок 14.52 показывает пример моделирования еще трех полосовых фильтров – на основе комбинаций линий передачи разного и на основе нескольких и одной схемы общего типа General Circuit Element. Под диаграммой модели представлены спектрограммы, демонстрирующие идентичность АЧХ этих фильтров. Рекомендуется детально познакомиться с окнами задания параметров элементов – прежде всего блока **General Circuit Element** в третьем варианте фильтра.

Еще один пример моделирования фильтров на отрезках линий передачи (обычных и микрополосковых) показан на рис. 14.53. На этот раз моделируется полосовой заграждающий фильтр. На спектрограмме представлены равномерный спектр входного сигнала (белого шума) и спектры сигналов на выходе фильтров, представляющие их АЧХ.

Представленные примеры дают достаточное представление о технике моделирования радиочастотных устройств диапазона СВЧ.

14.4.3. Моделирование многокаскадных радиочастотных систем

Современные радиотехнические устройства обычно являются многокаскадными системами. Отдельные каскады включаются последовательно и, как правило, согласуются выбором одинаковых волновых сопротивлений по всему тракту таких устройств. Это предотвращает нежелательные отражения сигналов и способствует получению заданных характеристик устройств. Однако из-за комплексного представления всех параметров устройств получение точного согласования в широкой полосе частот оказывается довольно проблематичным делом.

Средства пакета **RF Blockset** позволяют моделировать каскадные схемы, содержащие различные компоненты – линии передачи, усилители, миксеры и др. Рисунок 14.54 показывает пример моделирования трех таких устройств с различными наборами блоков и одного устройства на основе уже упомянутого универсального блока **General Circuit Element**.

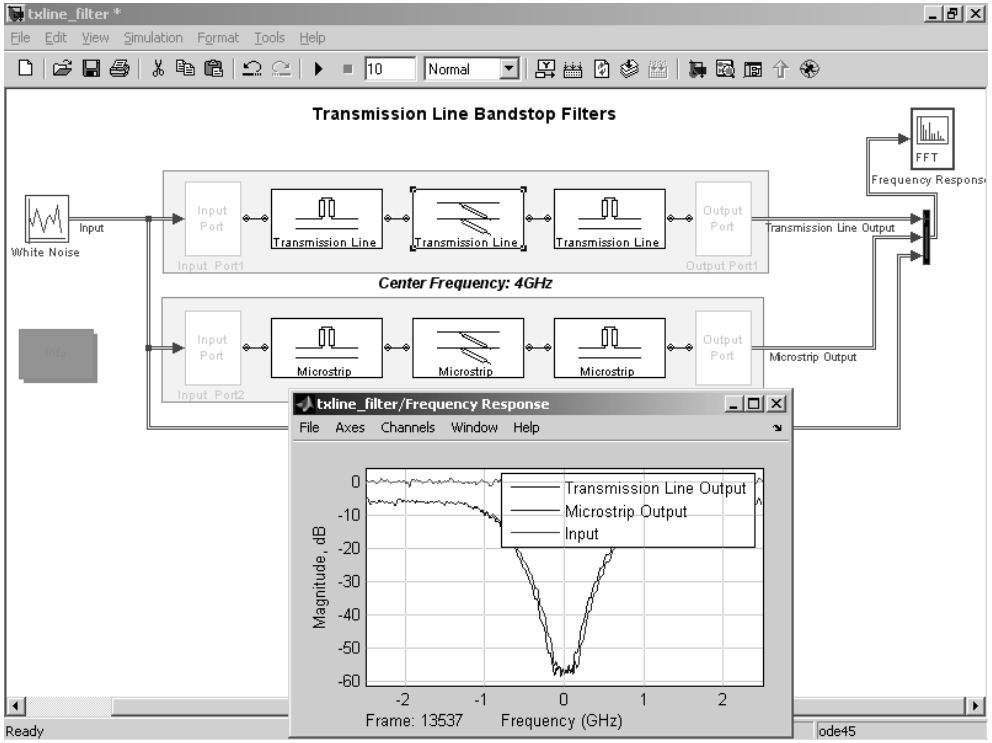


Рис. 14.53. Модель двух фильтров на отрезках линий передачи

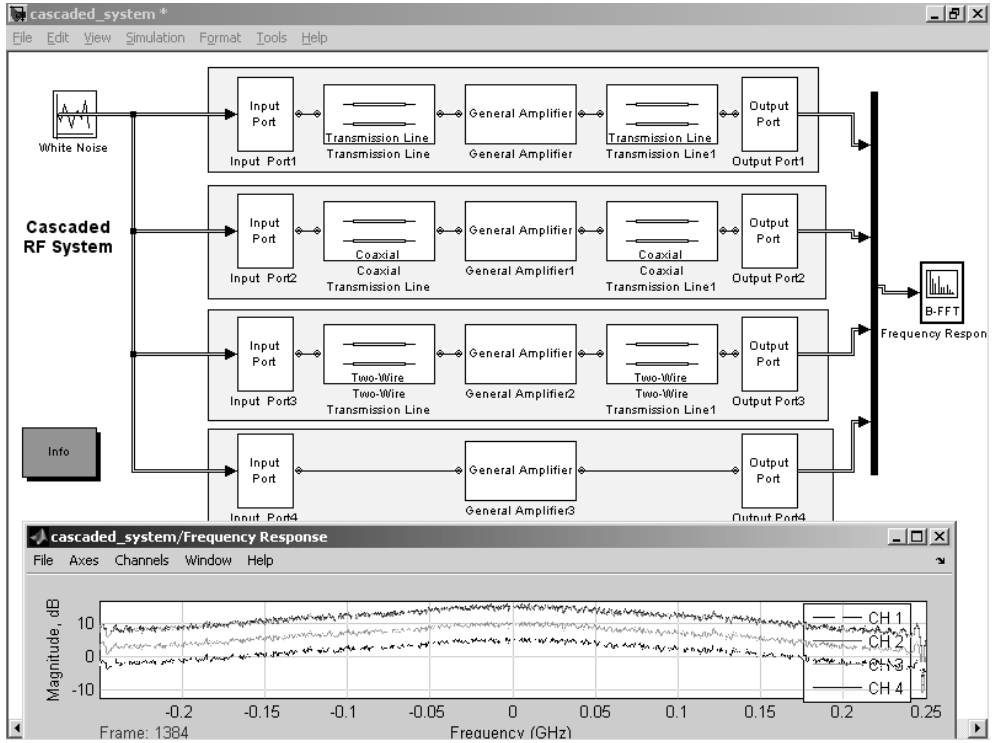


Рис. 14.54. Модель для ряда вариантов каскадных устройств

Частотные зависимости их коэффициентов передачи, полученные как реакции на белый шум, представлены под диаграммой модели. Для лучшего понимания работы этой модели рекомендуется внимательно просмотреть установки параметров всех блоков модели.

14.4.4. Примеры совместного применения пакетов RF и Communication Blockset

Как уже отмечалось, пакет расширения **RF Blockset** может использоваться с рядом пакетов расширений с близким характером применений. Выше уже были приведены примеры такого применения с пакетами **SP**, **DSP** и **RF Toolbox**. В справке по пакету **RF Blockset** представлена пара примеров совместного применения этого пакета с пакетом **Communications Blockset**.

Рисунок 14.55 демонстрирует работу тракта передачи сигнала с QPSK-модуляцией через усилитель с портами входа/выхода, построенный на основе блоков пакета **RF Blockset**. Передатчик и приемник сигнала используют средства пакета **Communications Blockset**. С помощью блоков графического представления сигналов из пакета **Communications Blockset** можно проследить, насколько искажается сигнал в ходе передачи в тракте усилителя.

Еще один пример совместного применения пакетов расширения **RF Blockset** и **Communications Blockset** представлен на рис. 14.56. Здесь анализируется передача сигнала с QAM-модуляцией (см. описание пакета **Communications Blockset** ниже) через тракт узкополосного усилителя с центральной частотой 2,1 ГГц. Анализатор спектра строит АЧХ приемника и канала передачи, а сигнал на выходе визуализируется с помощью глазковой диаграммы.

Разумеется, приведенные примеры иллюстрируют лишь часть обширных возможностей пакета **RF Blockset**. Однако это новейший пакет, впервые введенный в систему MATLAB 7 + Simulink 6. Пока в пакете можно обнаружить ряд «пенок», которые в основном связаны с нестыковкой форматов данных и параметров блоков этого пакета с блоками других пакетов расширения. Пакету явно недостает своих блоков генерации сигналов и контроля их параметров. Возможности пакета реализуются в основном в частотной области, и возможности анализа широкополосных и СВЧ-схем во временной области у пакета практически отсутствуют.

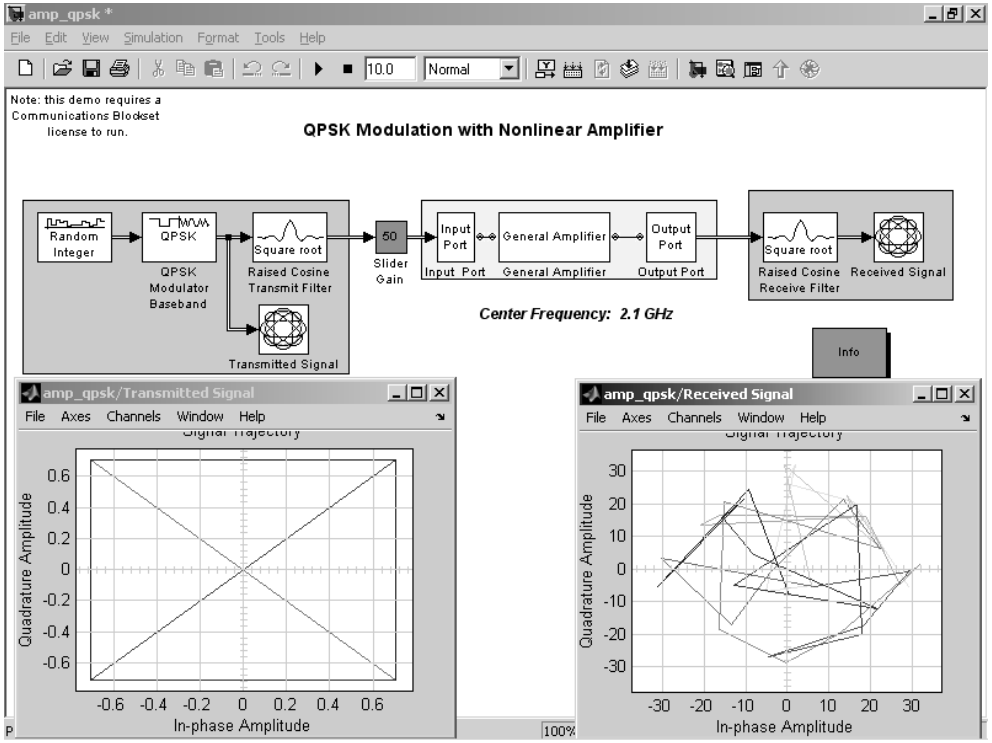


*Рис. 14.55. Модель тракта передачи сигнала с **QPSK-модуляцией***

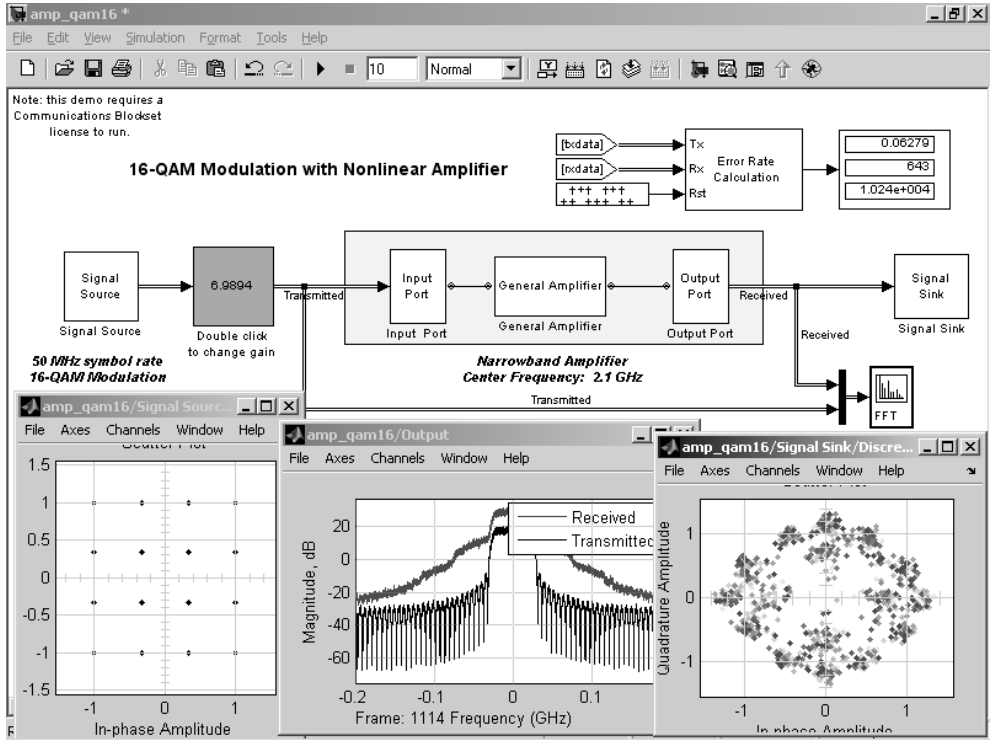


*Рис. 14.56. Модель тракта передачи сигнала с **QAM-модуляцией** и узкополосным каналом передачи*

Можно ожидать, что эти недостатки будут устранены в очередных реализациях данного весьма полезного пакета. Это очевидно, если учесть, что области применения СВЧ-техники непрерывно развиваются – достаточно отметить разработку СВЧ-микросхем беспроводных сетей, сотовых телефонов, кабельного и эфирного телевидения и многих, многих других устройств связи, коммуникаций и вычислительной техники.

14.5. Пакет Communications Blockset

14.5.1. Назначение пакетов Communications Blockset и Communications Toolbox

Пакеты расширения **Communications Blockset** и **Communications Toolbox** – одни из самых крупных пакетов расширения системы MATLAB, предназначенных для исследования, моделирования и проектирования коммуникационных систем и устройств. В рамках небольшого урока их подробное рассмотрение невозможно, и в связи с этим ограничимся общим обзором пакета **Communications Blockset** и несколькими примерами его применения.

Пакет **Communications Blockset** (последняя версия 3.5) представлен обширным набором моделей, включающих в себя описания важнейших коммуникационных элементов: источников сигналов с различными законами распределения, квантователей, модуляторов, демодуляторов, виртуальных устройств индикации и многих других устройств.

Пакет допускает совместную работу не только своих двух указанных выше частей, но и работу с функциями и компонентами, входящими в другие пакеты расширения системы MATLAB, в том числе описанные выше. Все это делает возможности системы MATLAB в области моделирования коммуникационных и связных устройств поистине уникальными и удивительными.

14.5.2. Основы работы

Будучи пакетом расширения системы MATLAB + Simulink, пакет **Communications Blockset** допускает три основных вида работы:

- визуально-ориентированная подготовка моделей и их исполнение в среде Simulink;
- применение функций пакета в командах, исполняемых в командном режиме MATLAB;
- применение функций пакета в составе m-файлов MATLAB.

Основным и наиболее удобным видом работы является визуально-ориентированная подготовка моделей путем перетаскивания компонентов мышью из окон библиотек в окно создаваемой модели коммуникационной системы с последующим соединением этих компонентов друг с другом. При этом используются все возможности пакета Simulink, включая имеющиеся в нем модели компонентов общего назначения – например, модели соединений между блоками, виртуальные осциллографы и прочие регистрирующие приборы. Полученная модель после установки ее параметров запускается, затем изучается и корректируется. Весьма практичным является подход, когда модель создается не «с нуля», а путем модификации одного из подходящих демонстрационных примеров.

В целом пакет рассчитан на самообучение пользователя в процессе работы с ним. Это достигается за счет включения в пакет множества заранее созданных моделей компонентов и довольно наглядных примеров их применения.

14.5.3. Доступ к библиотеке пакета и ее разделам

Для вызова окна с разделами библиотек пакета **Communications Blockset** достаточно в командном режиме MATLAB исполнить команду

```
>> commlib;
```

Появится окно, показанное на рис. 14.57 слева и сверху. В этом окне расположены значки с основными разделами библиотеки коммуникационных устройств и примеров их применения. Активизация того или иного значка (двойным щелчком) открывает окно соответствующего раздела библиотеки. В качестве примера на рис. 14.57 вокруг основного окна библиотеки показаны окна с рядом разделов библиотеки пакета. Каждый раздел содержит ряд окон с соответствующими блоками. Разумеется, вызов окна рис. 14.57 возможен и из браузера библиотек Simulink.

Библиотека пакета **Communications Blockset** содержит множество моделей компонентов, с помощью которых можно моделировать как самые простые, так и весьма сложные устройства и системы связи. Несмотря на обилие входящих в нее средств, не надо забывать, что они составляют лишь подмножество обширного набора средств моделирования, которые имеются в пакете Simulink. При этом обеспечиваются простой графический интерфейс пользователя и визуально-ориентированный способ подготовки моделей устройств и систем, полностью аналогичные применяемым в пакете Simulink.

Документация по пакету **Communications Blockset** (руководство пользователя и справочник по блокам) в формате PDF насчитывает более 800 страниц. В данном уроке ограниченного объема повторять ее не имеет смысла. Поэтому

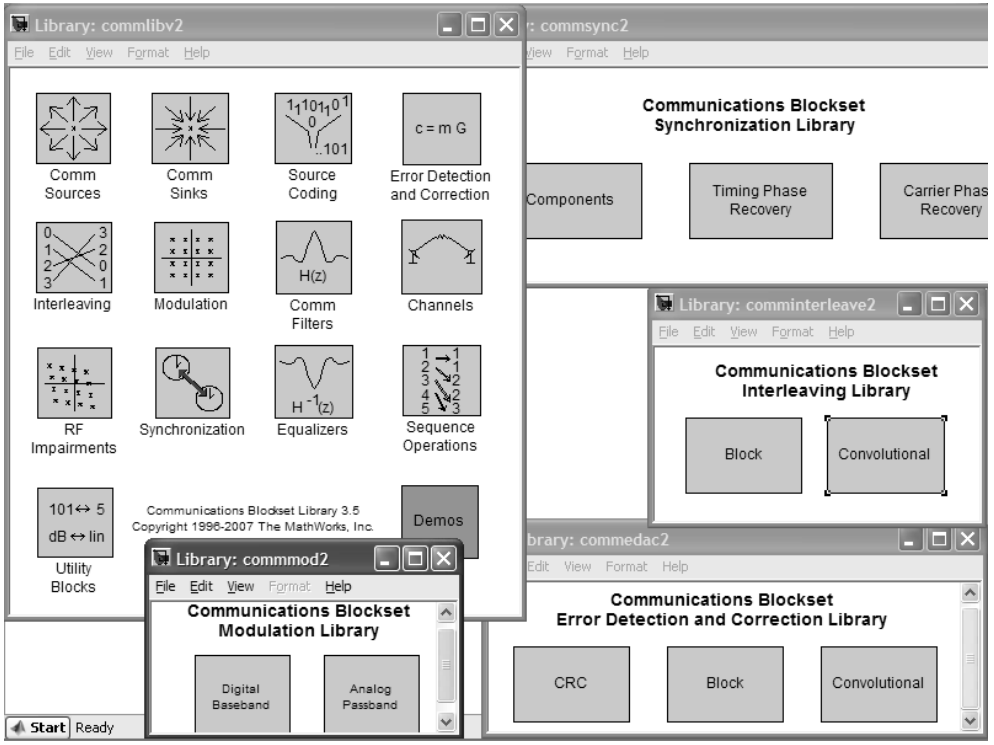


*Рис. 14.57. Окна библиотеки пакета **Communications Blockset** и нескольких ее разделов*

основное внимание уделено описанию того, как обеспечить доступ к тем или иным средствам пакета, а не самим средствам. Последние описаны выборочно.

Огромное число примеров применения тех или иных блоков пакета можно найти в справке по пакету, особенно в разделе демонстрации моделей (Demonstration Models). Обширный перечень примеров можно найти и в разделе **Demos** справки. Там же можно найти тематический и алфавитный каталоги всех блоков пакета.

14.5.4. Источники и получатели коммуникационных сигналов

Активизируя значки разделов библиотеки пакета по источникам сигналов и их получателям, можно вывести полный набор их блоков – рис. 14.58.

Для вызова окна установки параметров любого из представленных источников достаточно выполнить на значке объекта двойной щелчок. В пакете имеется представительный набор источников, представляющих модели реальных источ-

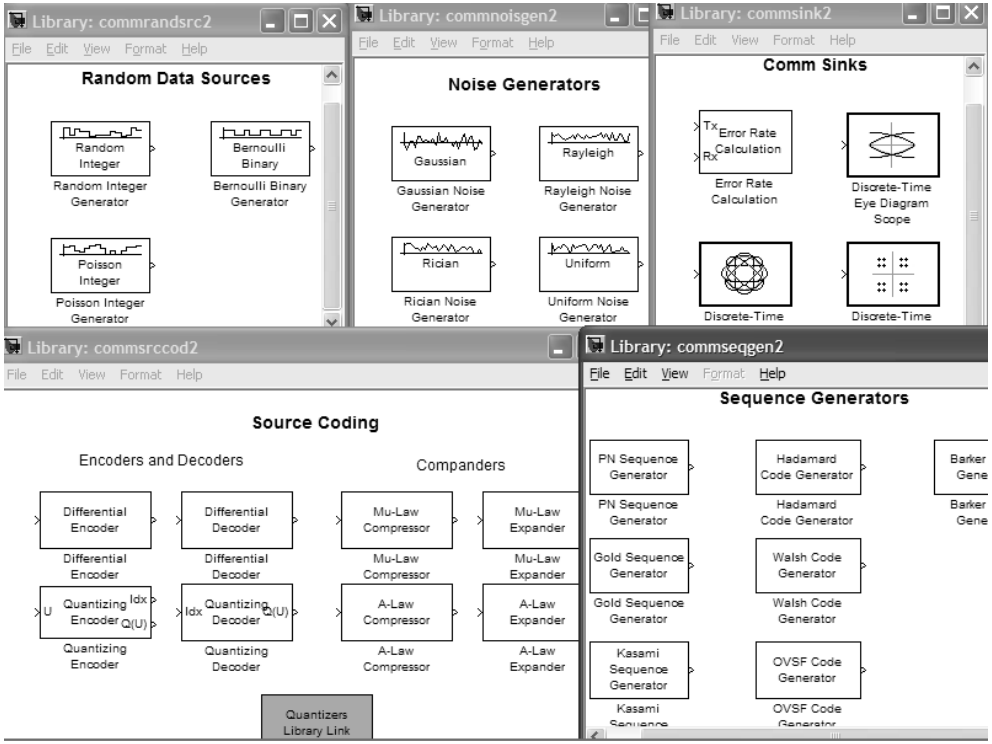


Рис. 14.58. Блоки источников и получателей сигналов

ников сигналов с шумом. Краткие названия источников приводятся под значками их моделей, а полное описание источников дается в окнах их параметров.

Большинство источников представлены их статистическими моделями. Ввиду общеизвестности законов распределения сигналов источников мы не будем обсуждать эти законы. Соответствующие расчетные формулы приведены как в литературе по статистической радиотехнике, так и в электронной справочной системе пакета и в его описании в формате PDF.

Для тестирования любого источника можно перетащить его блок в окно новой модели Simulink и подключить к источнику тот или иной виртуальный регистратор, например осциллограф и измеритель уровня сигнала. Пример этого представлен на рис. 14.59. Там же показаны осциллограммы сигнала и (справа) окно установки параметров источника – в данном случае генератора шума с гауссовым законом распределения.

Аналогичным образом можно задать любой другой источник, установить его параметры и опробовать в виртуальной работе. Читателю рекомендуется проделать это самостоятельно по образу и подобию описанного примера.

Библиотека блоков регистрации сигналов представлена на рис. 14.58 в верхнем правом углу. Примеры применения регистрирующих устройств можно найти

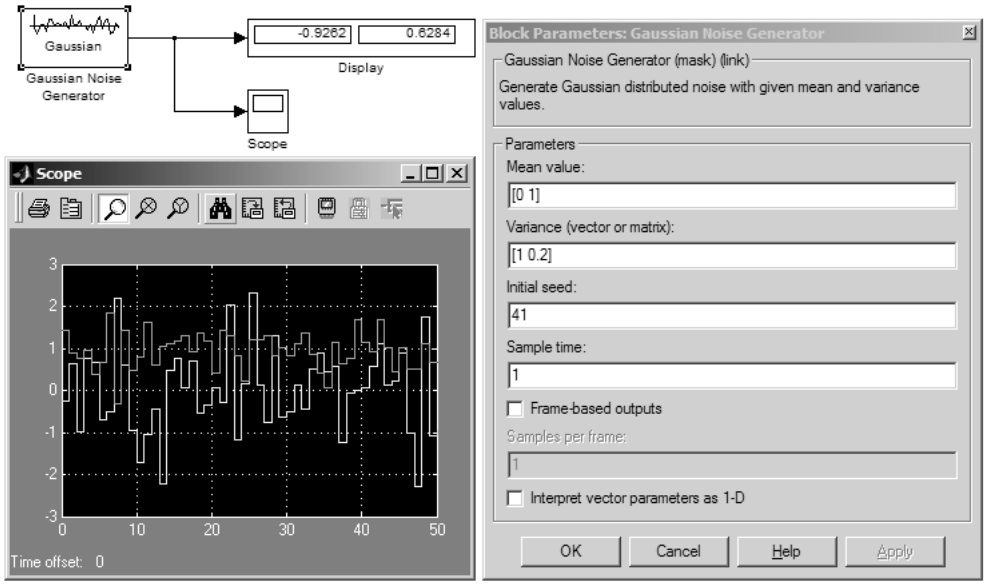


Рис. 14.59. Пример тестирования источника сигнала

в справке. На рис. 14.60 представлен пример **Eye-Pattern Demo**. Его окно выводится активизацией соответствующего значка в окне библиотеки. В нем, помимо модели, показан результат работы регистрирующего блока **Error Meter** (измеритель ошибок).

Работу регистратора **Eye-diagram/scatter plot** поясняет рис. 14.61. Нетрудно заметить, что строятся временная диаграмма (так называемая *глазковая диаграмма – eye diagram*; слева), распределение точек на плоскости (*диаграмма рассеяния – scatter plot*) и диаграмма соединений между точками. Графики такого рода широко используются во многих демонстрационных примерах описываемого пакета расширения. Мы рассмотрим их по мере описания этих примеров.

14.5.5. Моделирование кодирования и декодирования

Раздел библиотеки **Source Coding** содержит модели кодирующих и декодирующих устройств для источников сигнала. Большая их часть – это квантователи того или иного вида, обеспечивающие тот или иной способ преобразования аналоговой информации в дискретную и обратно. Окно этого раздела, представленное на рис. 14.58 справа внизу, дает представление об имеющихся моделях этих устройств.

Пример применения квантователя с постоянным по уровню шагом квантования представлен на рис. 14.62. Там же даны осциллограммы, поясняющие его ра-

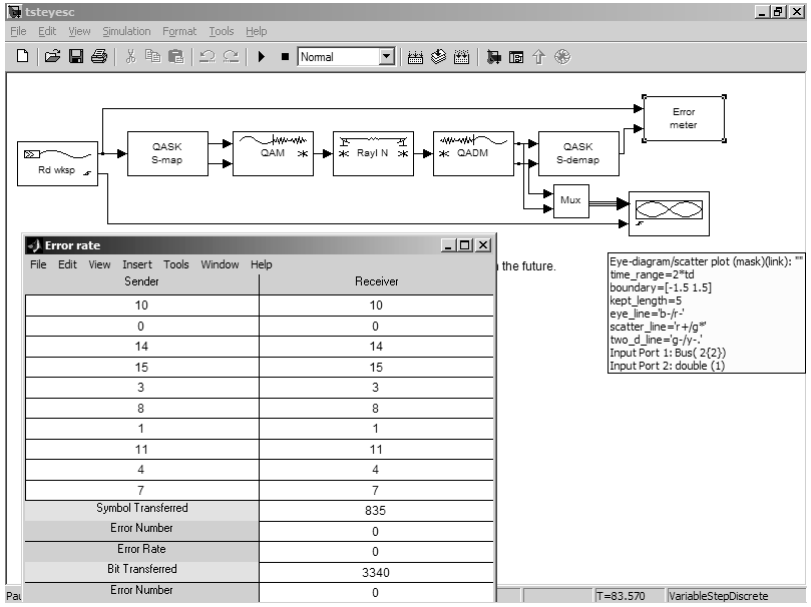


Рис. 14.60. Окно примера ***Eye-Pattern Demo***

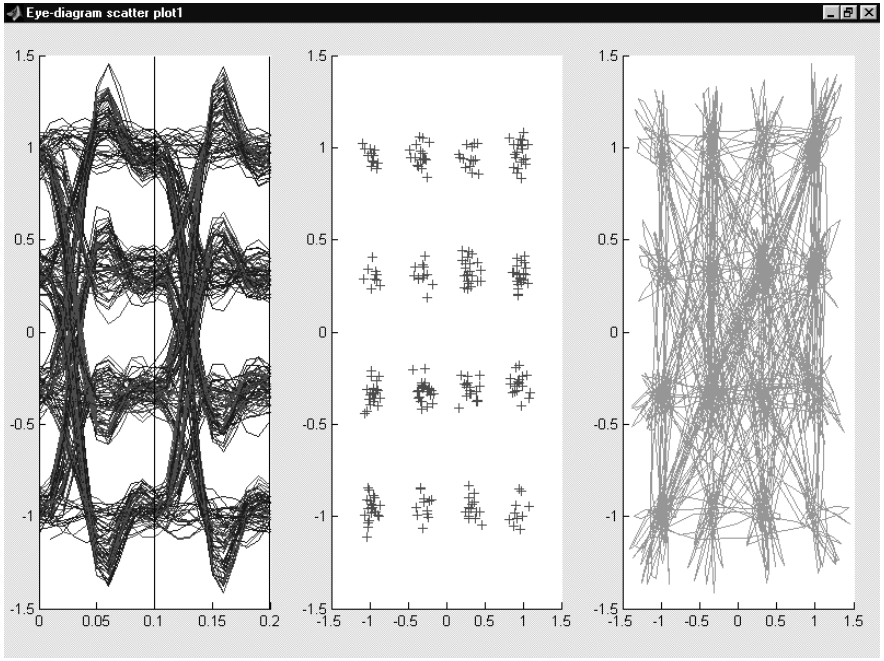


Рис. 14.61. Пример работы регистратора ***Eye-diagram/scatter plot***

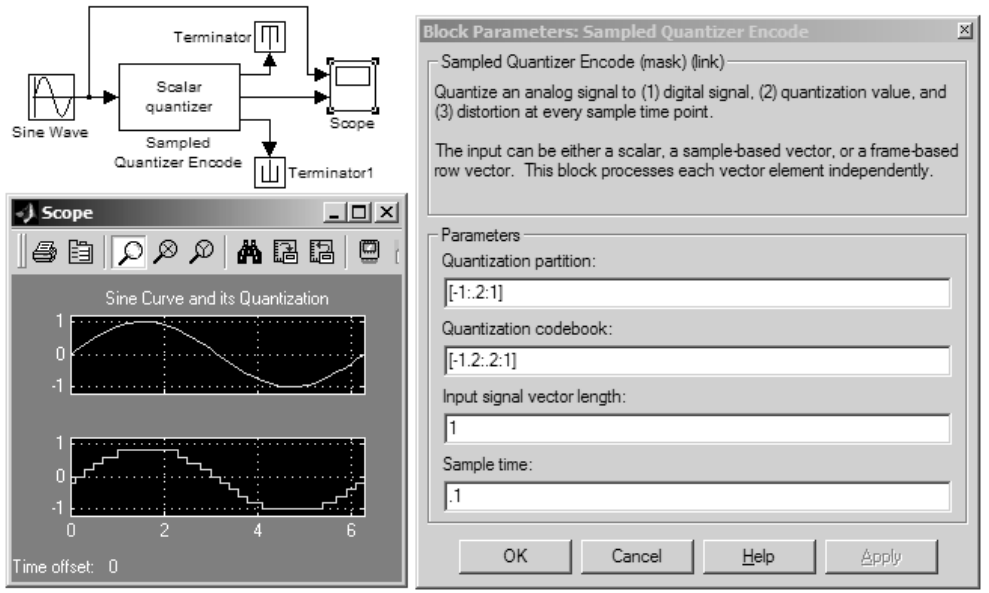


Рис. 14.62. Пример квантования синусоиды с постоянным шагом по уровню

боту, и окно установки параметров квантователя. Квантование такого рода принято называть дифференциальным, поскольку квантователь реагирует на приращение уровня сигнала относительно предшествующего уровня на заданную величину. Обратите внимание на то, что из трех выходов квантователя используется только один (квантованных сигналов), остальные выходы заглушены.

Еще один пример на работу квантователя с дифференциальной импульсно-кодовой модуляцией DCPM (Differencial Pulse Code Modulation) представлен на рис. 14.63. Здесь показаны процессы как кодирования, так и декодирования сигнала. Окна установки параметров DCPM квантователя и декодера, также показанные на этом рисунке, позволяют судить о простоте настройки этих блоков.

По образу и подобию приведенных примеров читатель может ознакомиться с другими типами кодирующих устройств и примерами их применения.

14.5.6. Моделирование модуляторов и демодуляторов

Модуляторы – пожалуй, самые распространенные устройства в технике связи. В этом нет ничего удивительного, поскольку именно модуляция (процесс наложения информации на несущее колебание) обеспечивает саму принципиальную возможность осуществления беспроводной связи. Поэтому библиотека блоков

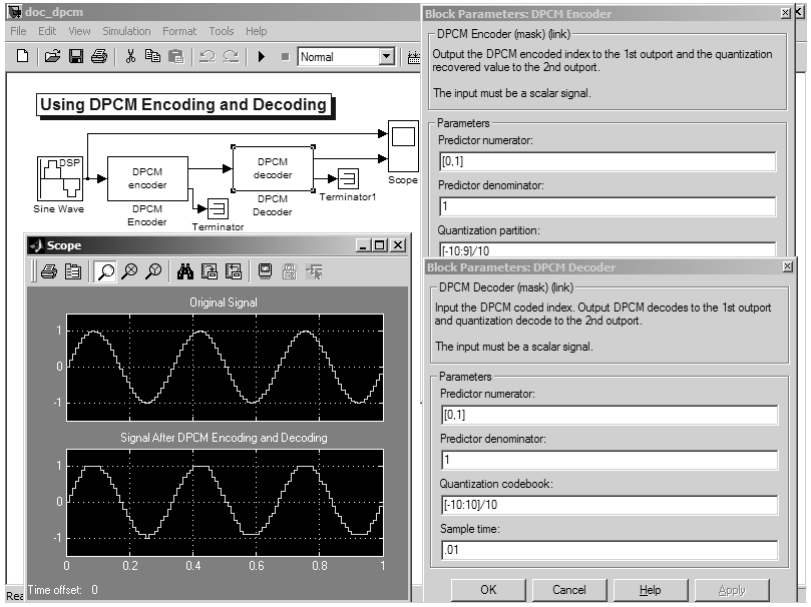


Рис. 14.63. Пример кодирования и декодирования сигнала с применением квантования

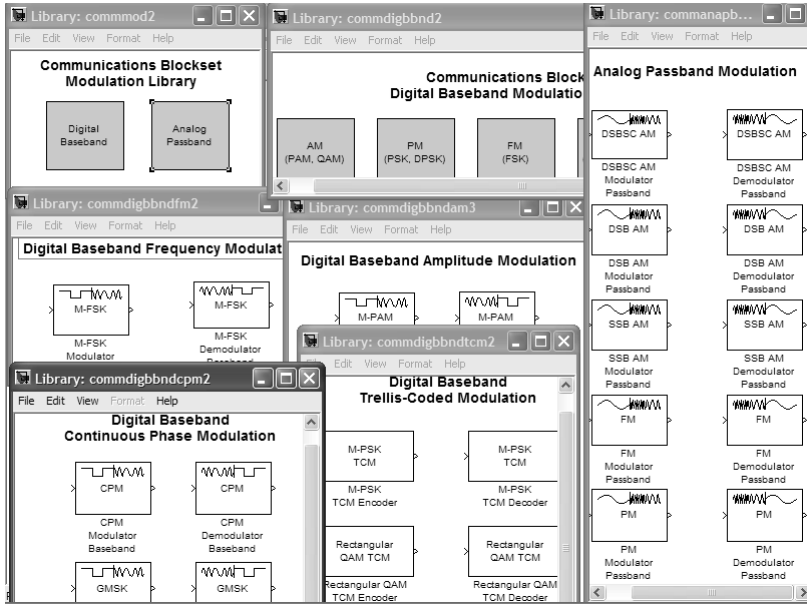


Рис. 14.64. Состав блоков модуляторов. Главное окно библиотеки модуляторов и окна ее разделов

модуляторов и демодуляторов в пакете **Communications Blockset** (рис. 14.64) способна поразить воображение даже специалиста в этих областях обилием предлагаемых средств и простотой их использования.

Для понимания названий блоков и примеров, основанных на применении модуляции/демодуляции, на рис. 14.65 представлена классификация видов модуляции сигналов, принятая в пакете **Communication Blockset**.

Отметим также наиболее распространенные виды импульсной модуляции:

- PWM (Pulse Wide Modulation) – широтно-импульсная модуляция;
- PTM (Pulse Time Modulation) – время-импульсная модуляция;
- CPM (Code Pulse Modulation) – кодо-импульсная модуляция.

Рисунок 14.66 показывает наглядный пример моделирования канала связи с MSK-модулятором и демодулятором. Под диаграммой модели представлены результаты моделирования в виде графика спектра и различных диаграмм. Рисунок 14.67 показывает диаграммы субблока регистрации.

Читатель может познакомиться с другими примерами моделирования модуляторов и демодуляторов самостоятельно, поскольку, как уже отмечалось, примеры имеются для всех моделей модуляторов, имеющихся в данной библиотеке.

14.5.7. Библиотеки каналов

Каналы включаются между передатчиком и приемником сообщений. Окно библиотеки каналов **Channels** (рис. 14.68) содержит всего четыре блока. Наименование реализуемых ими каналов указано под блоками.

На рис. 14.69 показан пример использования средств библиотеки каналов – моделируется бинарный канал с ошибками. Процесс моделирования иллюстрируется показателем регистратора ошибок и графопостроителя.

Обратите внимание на значок BCH code view table, расположенный под моделью канала. Его активизация открывает окно просмотра таблицы кодов. С другими примерами применения блоков этой обширной библиотеки читатель может ознакомиться самостоятельно.

14.5.8. Библиотека модулей синхронизации

Библиотека модулей синхронизации **Syncronization** представлена окном, показанным на рис. 14.70. Как нетрудно заметить, этот раздел библиотеки содержит четыре модуля фазовой автоподстройки частоты (ФАПЧ; английское сокращение PLL расшифровывается как Phase-Locked Loop). Обычно автоподстройка анализируется как типичная система инерционной обратной связи (инерционность создается фильтром нижних частот).

Пример применения модулей синхронизации показан на рис. 14.71. В этом примере, помимо демонстрации работы ФАПЧ, интересен многоканальный вывод сигналов (панель справа) с учетом фазовых сдвигов, вносимых всеми компонентами (их в этом примере шесть).

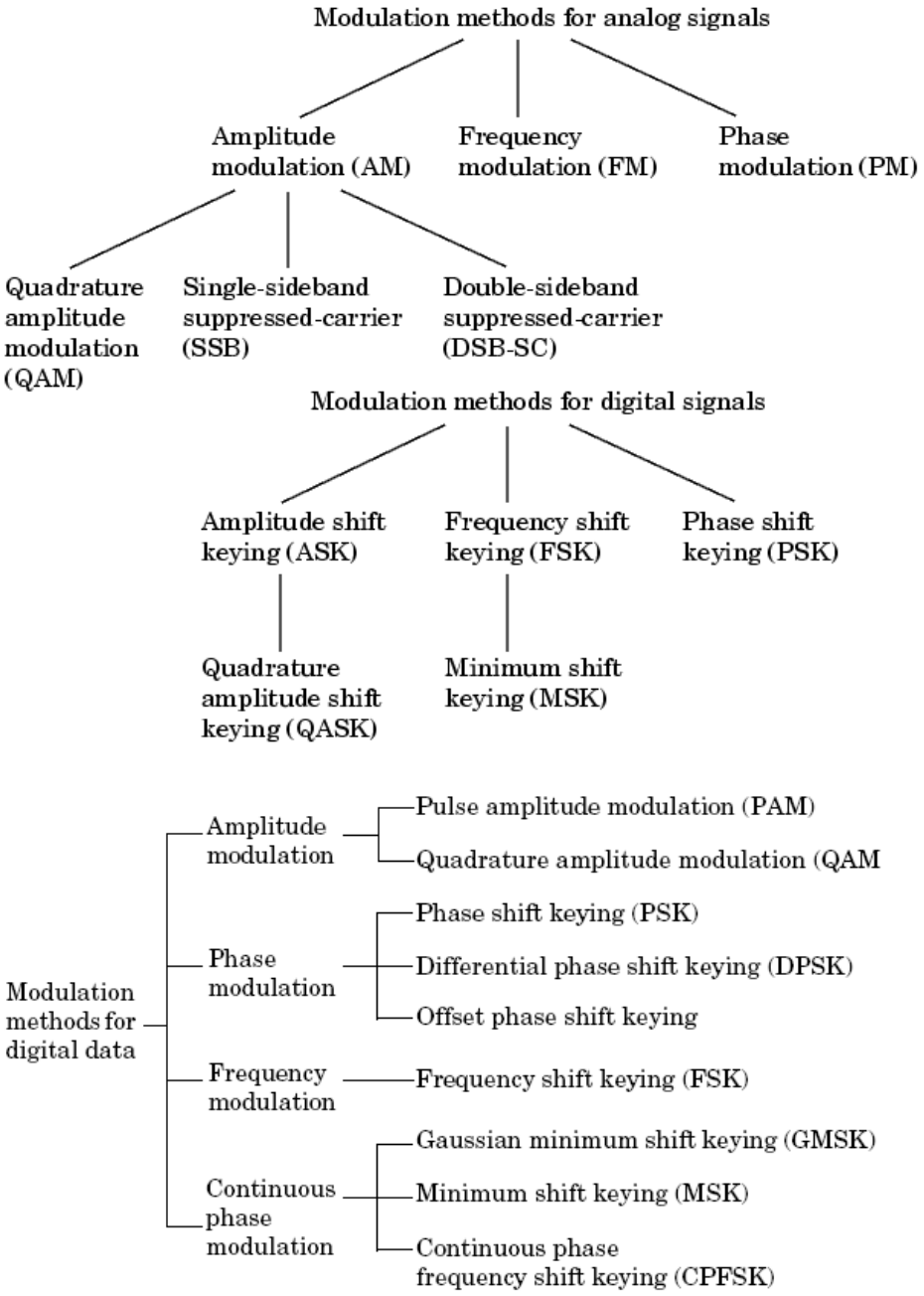


Рис. 14.65. Виды аналоговой (сверху) и цифровой (в середине) модуляций, а также относящиеся к ним методы (снизу)

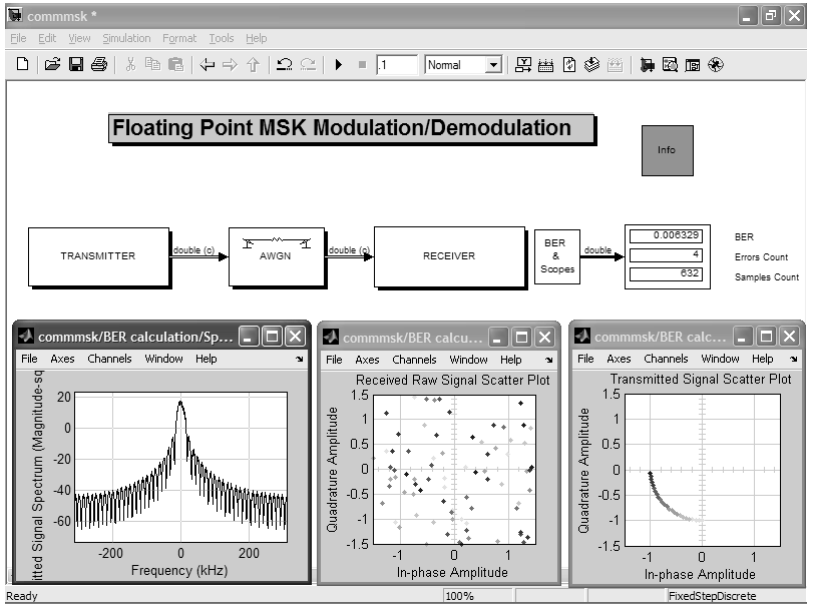


Рис. 14.66. Пример моделирования тракта с MSK-модуляцией

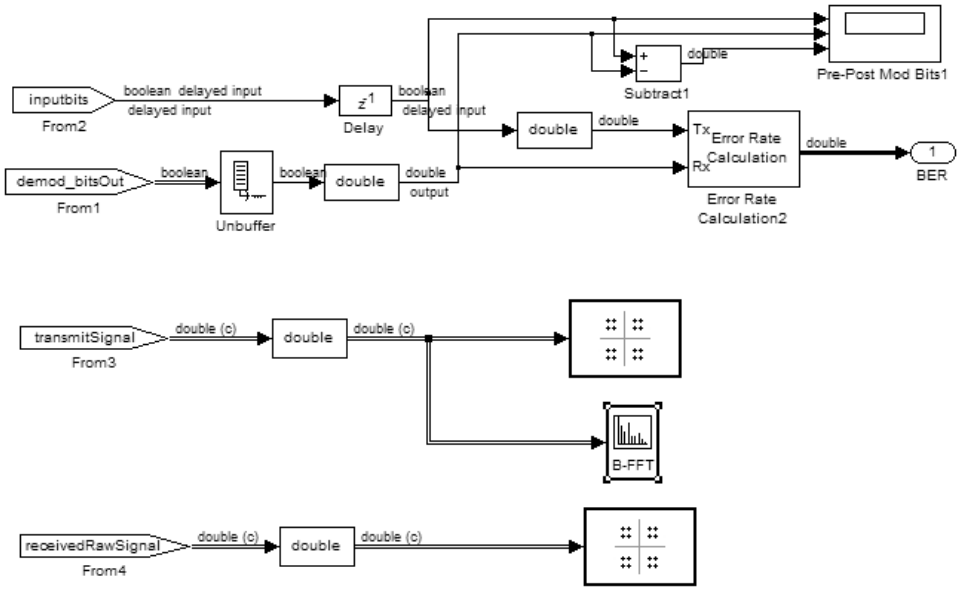


Рис. 14.67. Диаграммы субблока регистрации сигналов в тракте с MSK-модуляцией

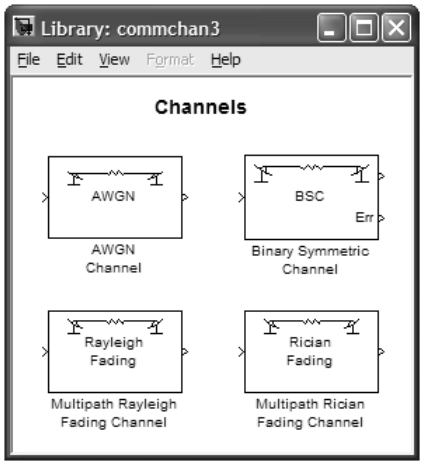


Рис. 14.68. Окно библиотеки каналов

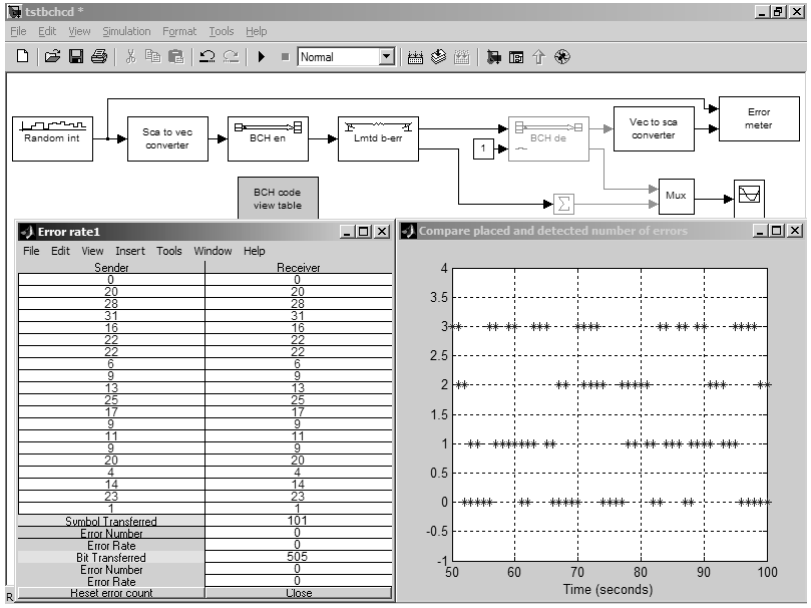


Рис. 14.69. Пример моделирования бинарного канала с ошибками

В справке по пакету можно найти множество достаточно сложных и больших примеров по применению модулей синхронизации. Большинство из диаграмм требуют для просмотра дисплей ПК с высоким разрешением и используют техни-

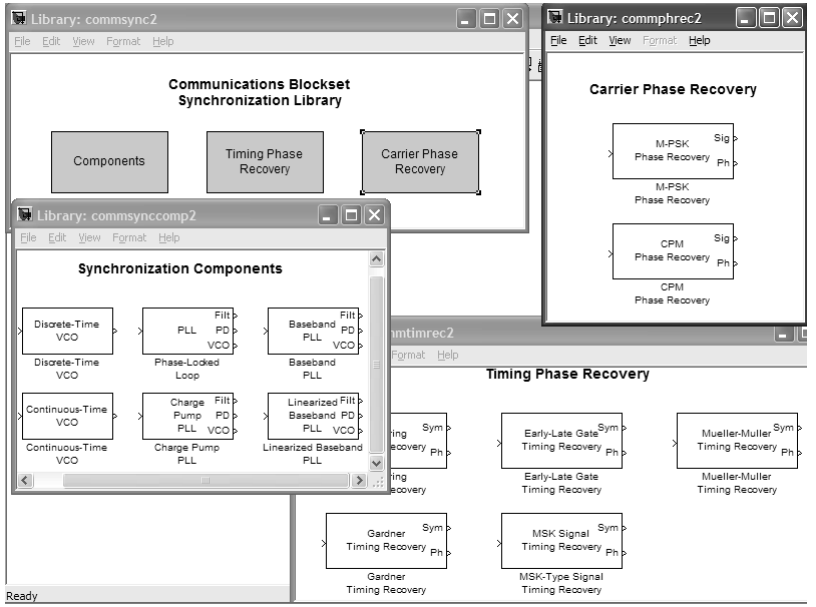


Рис. 14.70. Окно библиотеки модулей синхронизации

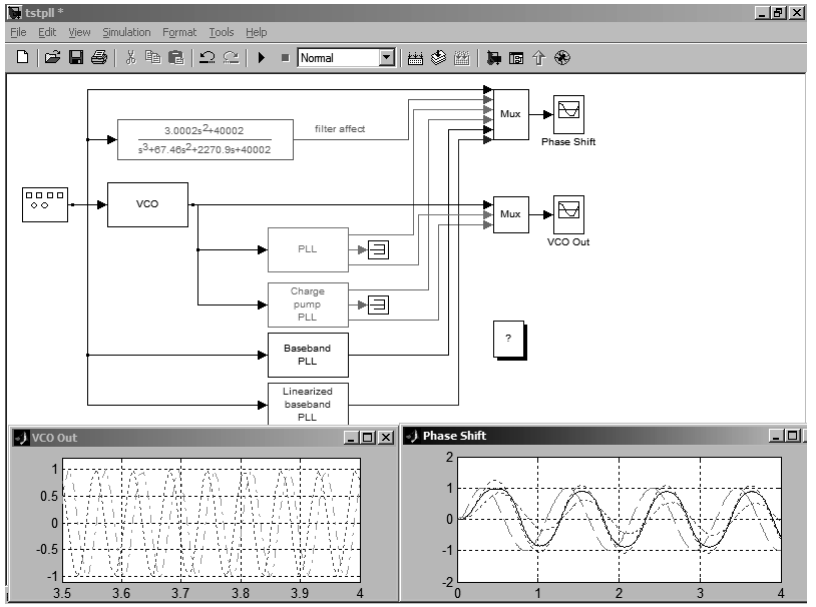


Рис. 14.71. Пример применения модулей синхронизации

ку красочной раскраски. Их представление в книге с черно-белыми рисунками невозможно. В связи с этим заинтересованному пользователю можно рекомендовать просмотр таких примеров на своем ПК.

14.5.9. Применение блоков детектирования ошибок и коррекции

При обработке, передаче и приеме цифровой информации большое значение имеют определение (детектирование) ошибок и их коррекция. Необходимые для этого средства сосредоточены в разделе библиотеки **Error Detection and Correction**. Полный состав блоков этой библиотеки представлен на рис. 14.72.

Этот раздел библиотеки состоит из трех разделов: **Convolutional (Свертка)**, **Block (Блок)** и **CRC**. Каждый из них имеет свой набор блоков в своем окне. Они и показаны на рис. 14.72.

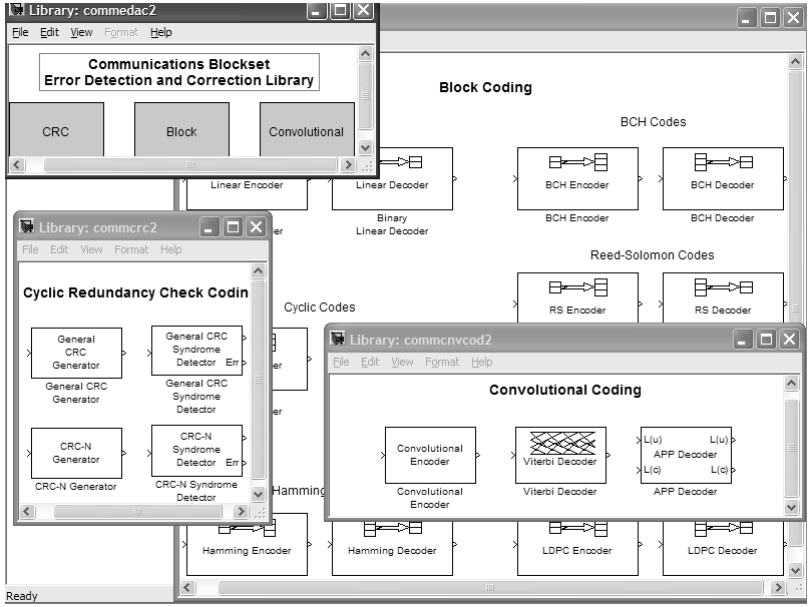


*Рис. 14.72. Состав раздела библиотеки **Error Detection and Correction***

В состав раздела **Convolutional** входит кодер Convolutional Encoder, основанный на операции свертки, и два декодера: декодер Витерби – Viterby Decoder и APP Decoder (APP происходит от слов «a posteriori probability»). Пример на применение первых двух названных блоков представлен на рис. 14.73. В нем показано также применение блока **Error Rate Calculator**, вычисляющего пределы ошибки после кодирования-декодирования случайного сигнала с распределением Бернулли.

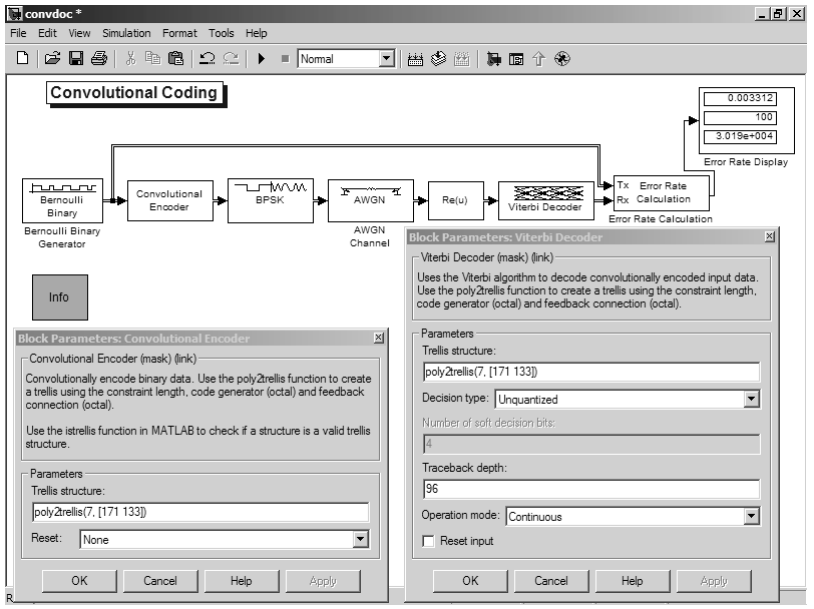


*Рис. 14.73. Примеры применения блоков **Convolutional Encoder** и **Viterby Decoder***

Пакет **Communications Blockset** позволяет создавать модели с встроенными подмоделями. Рисунок 14.74 дает пример построения такой системы для принятия «мягкого решения» – Soft-Decision Decoding. Один из блоков системы выполнен в виде подсистемы – его модель представлена под основной моделью.

Самый большой по набору блоков раздел библиотеки **Block Coding** содержит пары кодеров и декодеров для разного типа кодов:

- **Linear** – линейных;
- **Cyclic** – циклических;
- **Hamming** – Хаминга;
- **BCH** – специальный вид циклических кодов;
- **Reed-Solomon** – Рида Соломона (введены в реализацию пакета 2.1).

Рекомендуется просмотреть примеры применения этих блоков из справочных статей на них в справке.

14.5.10. Блоки фильтров и эквалайзеров

В проектировании коммуникационных устройств большое значение принадлежит фильтрации сигналов. Раздел библиотеки по блокам фильтров содержит основное окно и окна двух подразделов. Все они представлены на рис. 14.75.

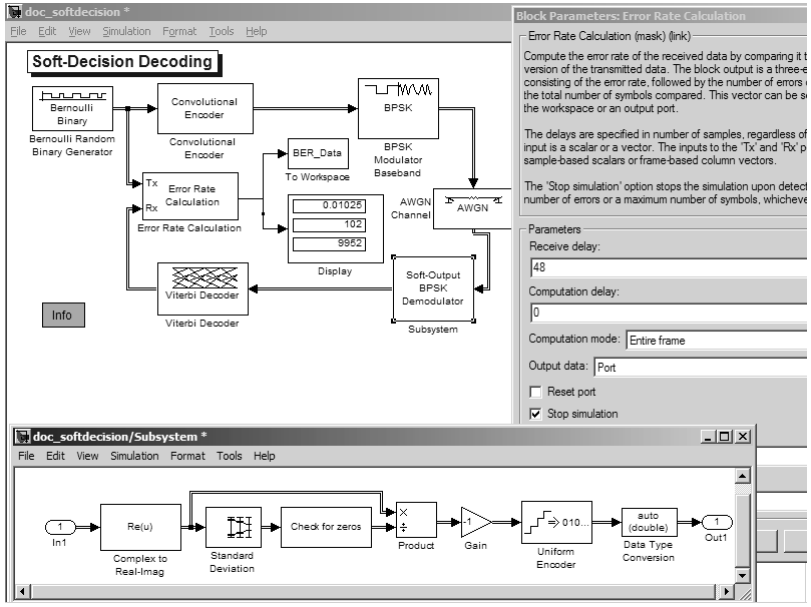


Рис. 14.74. Пример построения модели с подмоделью

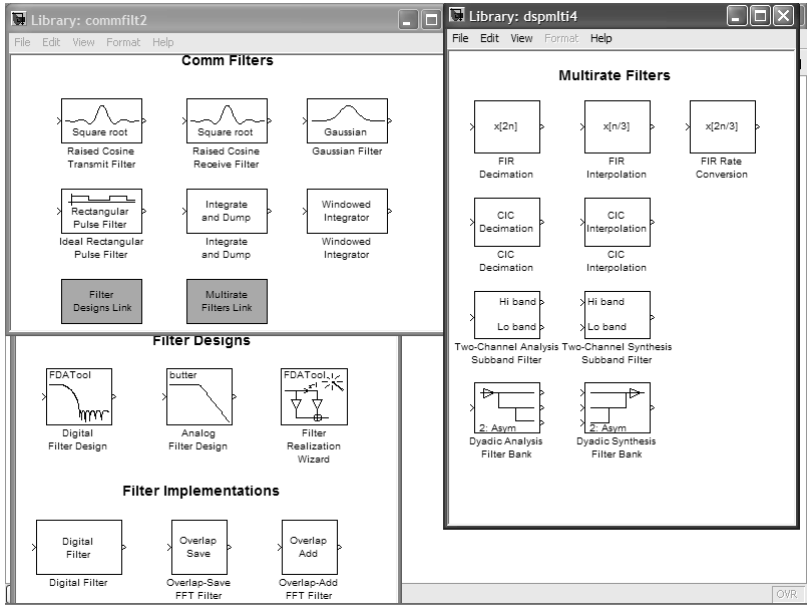


Рис. 14.75. Состав раздела библиотеки с блоками фильтров

В состав этого раздела включены блоки интеграторов. Рисунок 14.76 иллюстрирует работу цифрового интегратора Windowed Integrator, который интегрирует константу со значением +1 в течение времени (окна), равного 6 тактовым периодам. Нетрудно заметить, что в отличие от обычного (аналогового) интегратора сигнал на выходе цифрового интегратора нарастает линейно по ступенчатому закону. Окно установки параметров данного интегратора также показано на рисунке и в особых комментариях не нуждается. Отметим лишь выбор специфического параметра Integration Window Length (его значение взято равным 6), который задает длительность времени интегрирования.

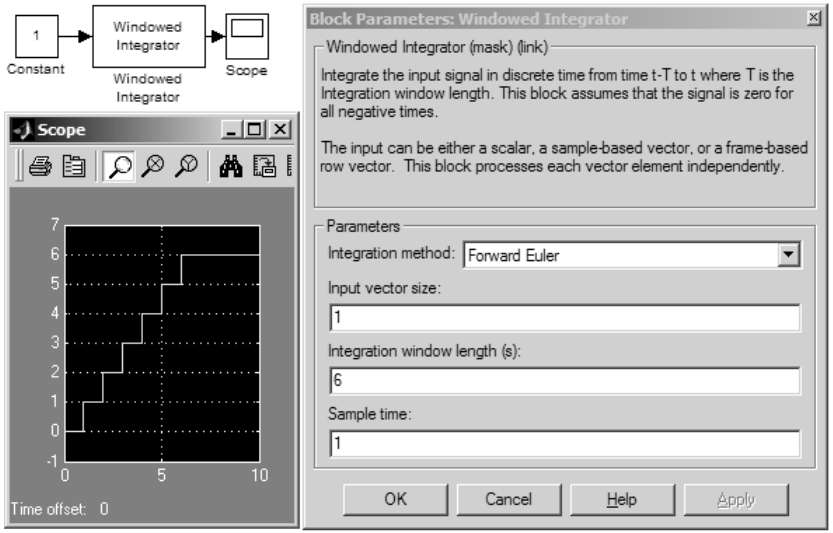


Рис. 14.76. Работа интегратора Windowed Integrator

Более необычными свойствами обладает интегратор Modulo Integrator. Он выполняет функции непрерывного интегратора с порогом, взятым по абсолютному значению. При достижении порога происходит сброс, и затем интегрирование подобным образом продолжается. При интегрировании константы блок вырабатывает пилообразный сигнал – рис. 14.77.

Работа интегратора Modulo Integrator при интегрировании синусоиды с амплитудой, равной 2, показана на рис. 14.78. Стоит присмотреться к осциллограммам сигналов на входе и выходе интегратора. Они дают наглядное представление о сути работы данного интегратора.

На рис. 14.79 показаны примеры построения каналов передачи QPSK vs. MSK. В верхней диаграмме используется фильтр с гауссовой АЧХ. Контроль передачи обеспечивается глазковой диаграммой.

К специальным типам фильтров со многими полосами относятся эквалайзеры. Набор блоков эквалайзеров показан на рис. 14.80.

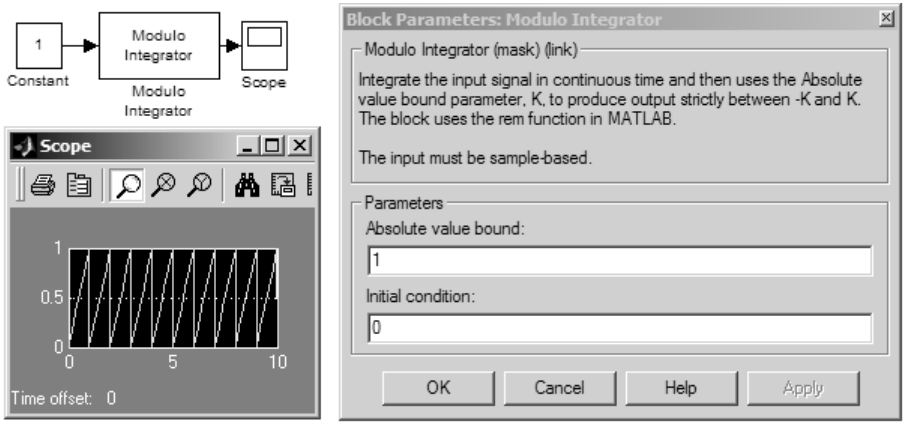


Рис. 14.77. Работа интегратора Modulo Integrator при интегрировании константы

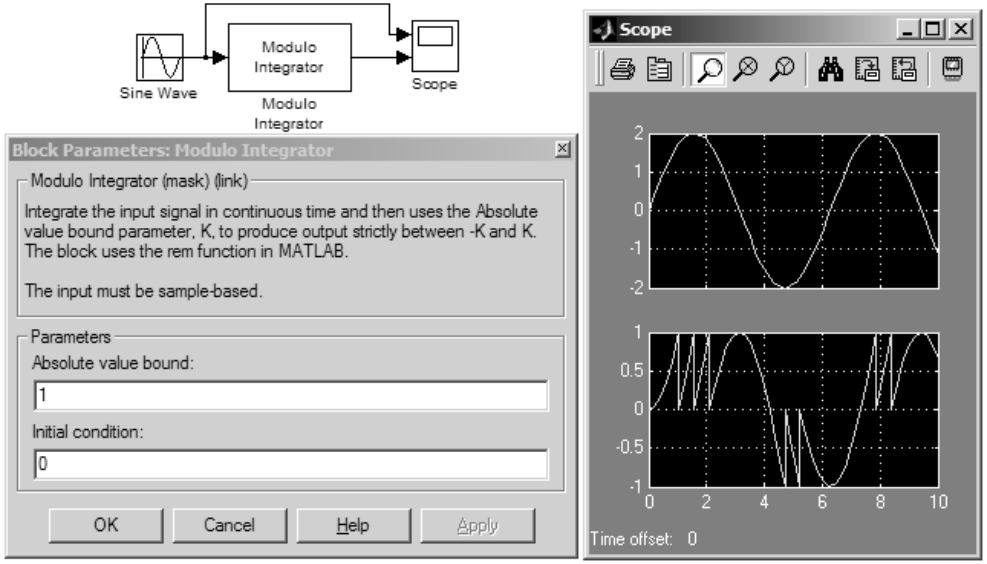


Рис. 14.78. Работа интегратора Modulo Integrator при интегрировании синусоиды

На рис. 14.81 показана модель адаптивного эквалайзера. Она построена на основе блока LMS-эквалайзера, использующего метод наименьших квадратов LMS. Подобные эквалайзеры обеспечивают эффективное выделение сигнала из шума путем автоматической настройки частот.

Помимо метода LMS, эквалайзер может использовать методы:

- Recursive Least-Square (RLS);
- Constant Module Algorithm (CMA).

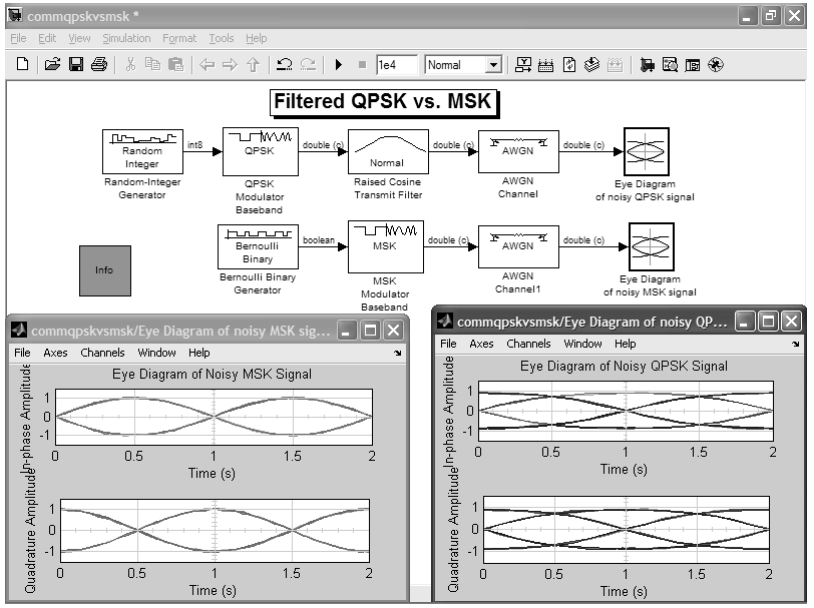


Рис. 14.79. Модель каналов передачи QPSK vs. MSK

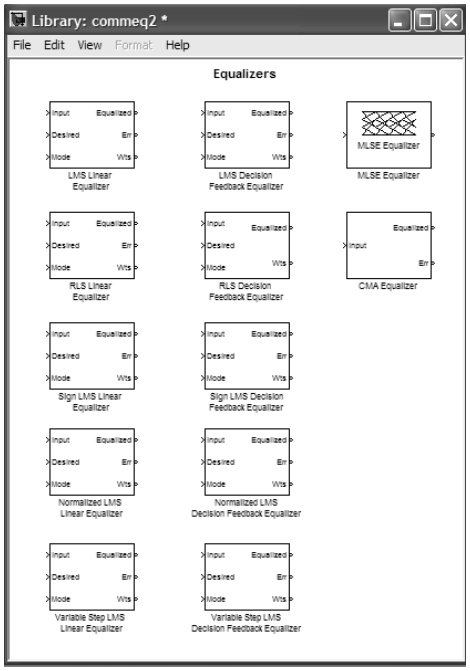


Рис. 14.80. Набор блоков эквалайзеров

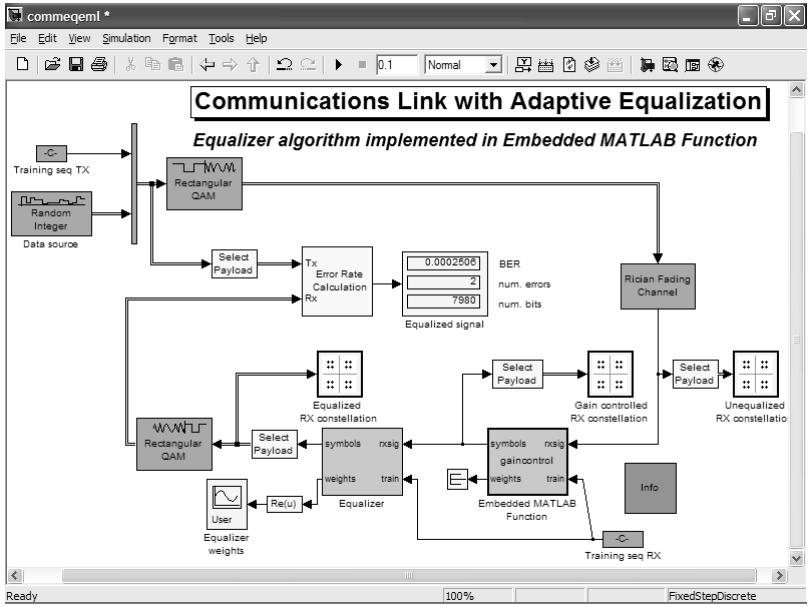


Рис. 14.81. Модель адаптивного эквалайзера

Для смены метода используется бирка initial Setting. Детали работы эквалайзера и список литературы по адаптивным фильтрам, к которым он принадлежит, можно найти в справке по этому примеру. Особенно интересно решена в этом примере задача визуализации работы активного эквалайзера с помощью блока из ряда осциллографических индикаторов.

14.5.11. Обзор других разделов библиотеки Communication Blockset

Блоки **Interleaving** служат для перестановок кодов (символов) в последовательностях блоков данных в соответствии с теми или иными правилами. Часть библиотеки **Interleaving** содержит два раздела: **Block** (перестановки в блоках) и **Convolutional** (перестановки при свертке). Эти разделы и окна с их блоками показаны на рис. 14.83.

Утилиты и функции этой библиотеки предоставляют большие возможности в обработке сигналов. Их применение можно найти в ряде примеров применения пакета, размещенных в разделе **Demos** справки.

Раздел библиотеки **Sequence operators** содержит блоки с операторами последовательностей (рис. 14.84 слева). Наконец, для различных дополнительных операций служат блоки последнего раздела библиотеки пакета – **Utility Blocks**. Их состав также показан на рис. 1.84, справа.

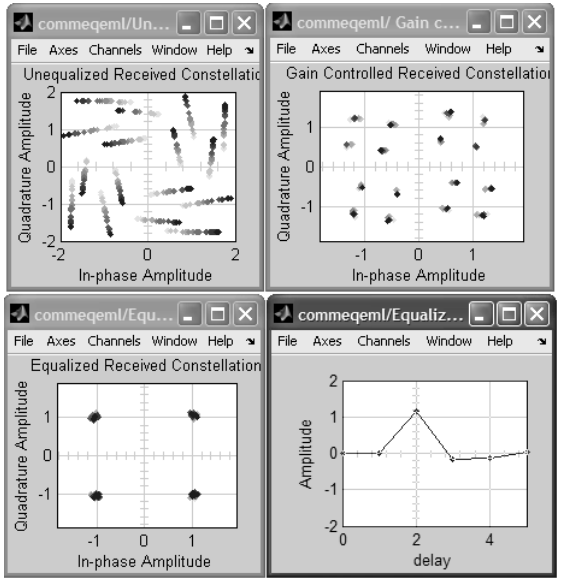


Рис. 14.82. Визуализация работы модели адаптивного эквалайзера

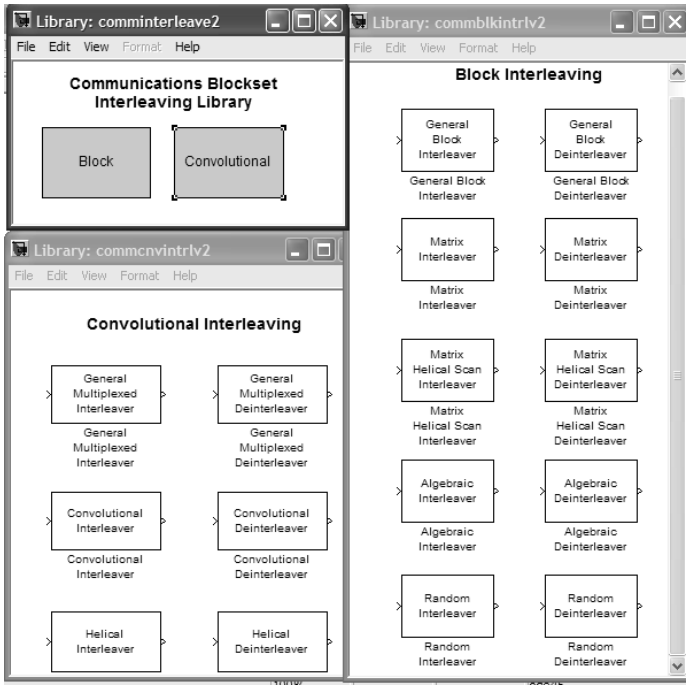


*Рис. 14.83. Состав раздела библиотеки **Interleaving***

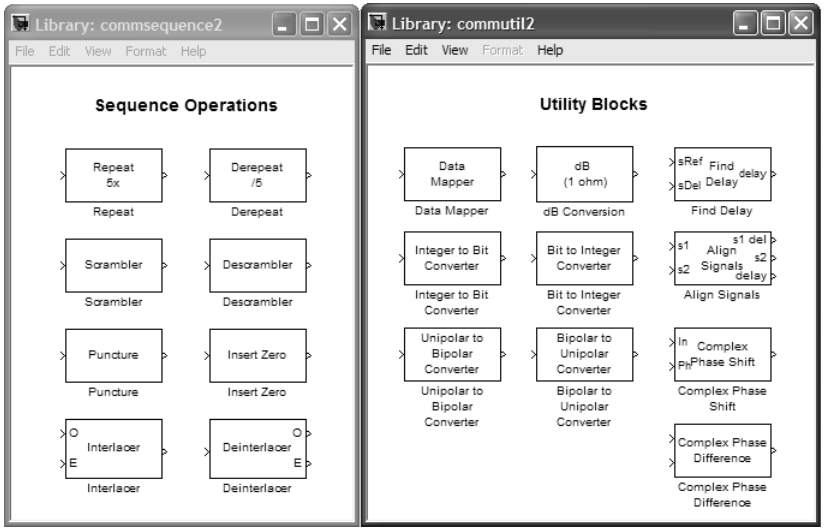


*Рис. 14.84. Состав разделов библиотеки **Sequence** и **Utility Blocks***

В разделе **Sequence** ряд блоков служит для задания и изменения последовательностей. Обычно они управляются бинарной последовательностью. Например, блок **Puncture** служит для изменения исходной последовательности, заданной вектором, путем указания в бинарном виде оставляемых (1) или удаляемых (2) элементов входного вектора. Управляющий вектор задается в окне параметров. Пусть входная последовательность есть [1:6], а вектор блока **Vector Puncture** – [110101]. Таким образом, из входной последовательности будут удалены числа 3 и 5 – рис. 14.85.

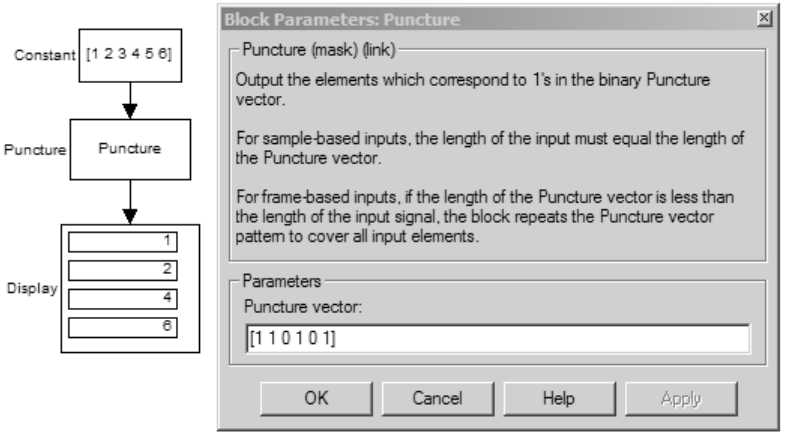


*Рис. 14.85. Работа блока **Puncture***

С другими блоками этой группы можно ознакомиться самостоятельно и по справке. Напоминаем, что если блок выделен, то команда **Help** контекстного меню правой клавиши мыши выводит окно справки по этому блоку. В этом окне, помимо детального описания блока, содержится, как правило, пример его применения.

В разделе **Utility Blocks** представлено довольно разношерстное семейство блоков. С их набором можно ознакомиться по названиям моделей.

14.6. Знакомство с Video and Image Processing Blockset

14.6.1. Инсталляция и назначение Video and Image Processing Blockset

Новейший пакет расширения **Video and Image Processing Blockset** впервые появился в выпущенной в сентябре 2004 г. системе MATLAB 7 Service Pack 1, причем к октябрю-месяцу вышла его улучшенная версия Video and Image Processing Blockset 1.0.1. В следующей версии Video and Image Processing Blockset 1.1 добавлено 10 новых блоков и 3 новых демонстрационных примера. Последней на момент подготовки данной книги была версия Video and Image Processing Blockset 2.3, входящая в MATLAB R2007b.

Пакет **Video and Image Processing Blockset** обеспечивает:

- создание и применение 2D-фильтров (FIR, медианного и свертки), 2D-преобразования (FFT, DCT, Hough) и геометрические преобразования (вращение, перемещения, разрезка изображений и др.);
- выполнение операций анализа изображений с применением усовершенствованных алгоритмов выделения кромок изображения, пороговых и морфологических преобразований, статистической обработки изображений;
- применение стандартных цветных изображений и видеопотоков и их обработку с использованием различных цветовых преобразований;
- моделирование и автоматическую генерацию кодов на языке С с использованием данных с плавающей и фиксированной точкой;
- оценка статуса видеопотоков в реальном масштабе времени и их просмотр с помощью соответствующих блоков.

Следует отметить, что пакет **Video and Image Processing Blockset** по существу повторяет многие возможности пакета **Image Processing Toolbox** и использует те же демонстрационные файлы рисунков и даже примеры применения. Пакет обеспечивает эффективную поддержку как средств цветной графики, так и видеосредств, обеспечивающих работу с видеофайлами. Он имеет собственные базовые примитивы и существенно усовершенствованные алгоритмы для обработки и представления изображений и видеофильмов. Это открывает обширные возможности применения этого пакета в художественной и научной графике, авиации, автомобилестроении, медицине, образовании и других областях.

14.6.2. Доступ к библиотеке блоков пакета

Доступ к библиотеке блоков пакета **Video and Image Processing Blockset** осуществляется, как обычно, из окна библиотеки пакета расширения Simulink 6. На рис. 14.86 показано основное окно библиотеки, содержащее 11 разделов библиотеки, пиктограмму демонстрационных примеров **Demos** и пиктограмму информационного окна **Info**.

Применение блоков библиотеки **Video and Image Processing Blockset** ничем не отличается от такового для диаграмм моделей основного пакета расширения системы MATLAB – Simulink.

Библиотека пакета **Video and Image Processing Blockset** содержит следующие разделы:

- **Sources** – блоки импорта источников изображений и видеопотоков;
- **Skins** – блоки регистраторов (средств экспорта и отображения изображений);
- **Analtsis & Enhancement** – блоки анализа и расширенных операций над изображениями и видеопотоками;
- **Conversions** – блоки конверсии изображений, например изменения гаммы цветов;

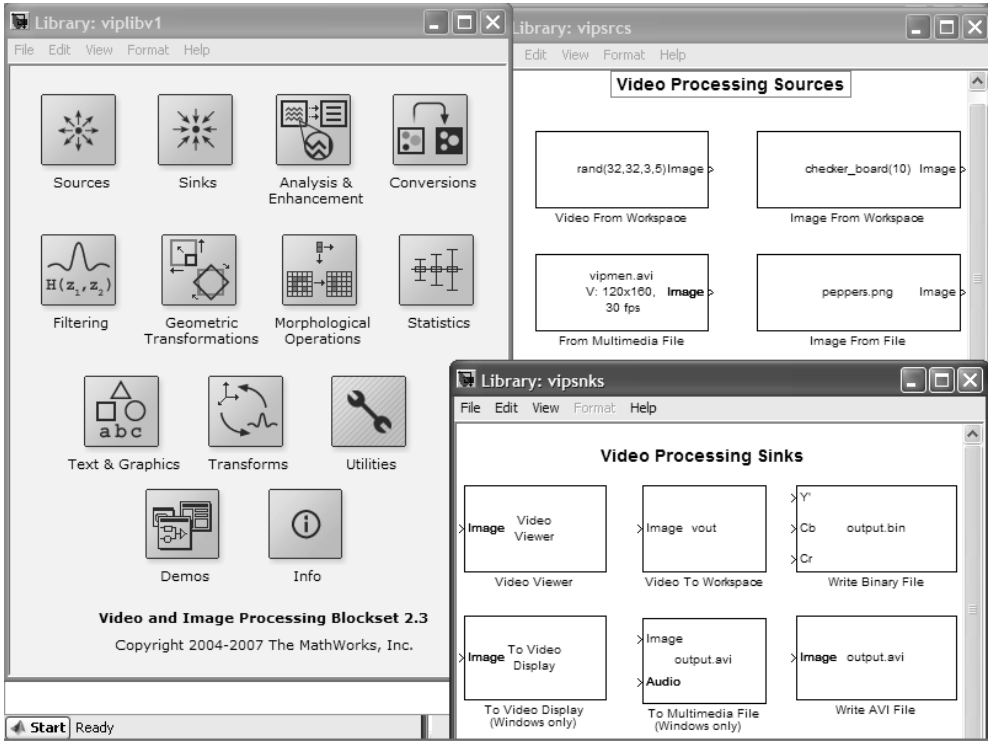


Рис. 14.86. Основное окно библиотеки блоков пакета ***Video and Image Processing Blockset***

- **Filtering** – блоки фильтрации изображений и видеопотоков;
- **Geometric Transformations** – блоки геометрических преобразований;
- **Morphological Operations** – блоки морфологических операций;
- **Statistics** – блоки статистической обработки;
- **Text & Graphics** – блоки вывода текстов и графики;
- **Transforms** – блоки преобразований изображений и видеопотоков, например изменения размеров и положения;
- **Utilities** – утилиты.

14.6.3. Поддерживаемые типы изображений и данных

Пакет **Video and Image Processing Blockset** поддерживает следующие типы изображений:

- бинарные (binary) – пиксели матриц которых имеют значения 0 или 1;
- полутоновые (intensity или greyscale) – пиксели матриц которых имеют ряд значений и представлены в форматах файлов с расширениями .tif, .png и др.;
- RGB – полноцветовые изображения, представленные тремя матрицами R, G и B красного, зеленого и голубого цветов;
- R'G'B' – полноцветовые сжатые изображения, представленные тремя матрицами R', G' и B' красного, зеленого и голубого цветов (файлы с расширением .jpg);
- видеопотоки – изображения, состоящие из ряда фреймов (кадров) и обычно представленные AVI-файлами.

Типы изображений и типы данных пакета **Video and Image Processing Blockset** соответствуют описанным ранее для пакета **Image Processing Toolbox**. В частности, большинство блоков поддерживает следующие форматы данных:

- с плавающей точкой двойной точности;
- с плавающей точкой одинарной точности;
- с фиксированной точкой;
- комплексные числа.

Особое внимание уделено поддержке формата с фиксированной точкой, обеспечивающего повышенную скорость вычислений. На ряде блоков в окнах установки их параметров предусмотрена отдельная вкладка **Fixed-point** для установки параметров работы с данными в этом формате.

14.6.4. Первый пример – просмотр видеофильма

В простоте работы с пакетом **Video and Image Processing Blockset** можно убедиться уже из первого примера его применения – просмотра видеофильма. Для этого нужно всего два блока – блок источника видеопотока (видеофильма) и блок

просмотра видео. Перетащив эти блоки в окно модели Simulink 6 и соединив их тремя соединениями, получим нашу первую модель, показанную на рис. 14.87.

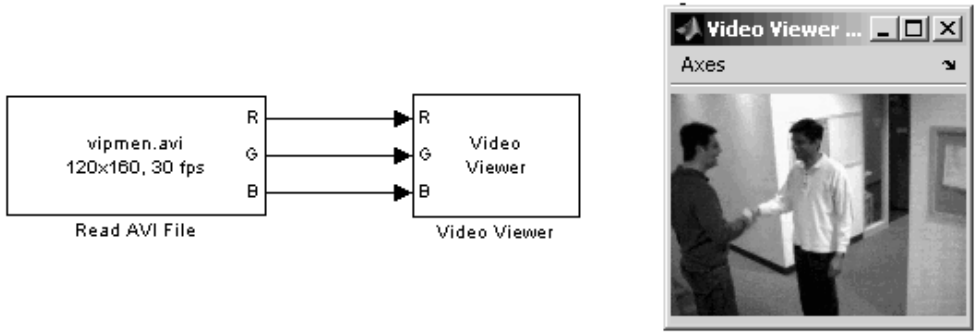


Рис. 14.87. Простая модель просмотра видеофильма

Источник видеопотока представлен блоком считывания файлов с расширением AVI – **Read AVI File**. По умолчанию блок настроен на считывание короткого демонстрационного AVI-файла vipmen.avi, но можно выбрать любой из имеющихся файлов. Блок имеет три выхода в формате RGB, обозначенных как R, G и B и соответствующих цветовым компонентам красного (read), зеленого (greed) и голубого (blue) цветов. Для просмотра видеофайлов служит блок **Video Viewer**, имеющий три входа R, G и B. Естественно, что их надо соединить с одноименными выходами блока **Read AVI File**. Правила соединения обычные для расширения Simulink 6.

Для правильной работы созданной модели надо проверить в окне установки параметров Simulink 6 следующие установки: Stop time=20, Type=Fixed-step и Solver=discrete. Теперь, пустив Simulin 6 нажатием кнопки пуска (треугольник), можно увидеть окно просмотра видеофильма, показанное на рис. 14.87 справа.

14.6.5. Блоки источников и получателей изображений

Приступим к детальному знакомству с блоками различных разделов библиотеки пакета. Источники изображений и видеопотоков широко используются при моделировании устройств и систем для обработки изображений и видеопотоков. В разделе библиотеки **Sources (Источники)** представлены пять блоков (рис. 14.86):

- **From Multimedia File** – считывает видеофреймы и/или сжатые мультимедийные звуковые файлы;
- **Image From File** – импортирует изображения из файлов изображений;
- **Image From Workspace** – импортирует изображения из рабочего пространства системы MATLAB;

- **Read AVI File** – считывает несжатые видеофреймы из AVI-файлов;
- **Read Binary File** – считывает бинарные видеоданные из файла;
- **Video From Workspace** – импортирует видеосигналы из рабочего пространства системы MATLAB.

Набор блоков этого раздела ограничен – представлены лишь блоки, считывающие изображения и видеопотоки из файлов или рабочего пространства MATLAB.

Раздел блоков представления изображений (экспорта видеоданных) представлен восемью блоками (рис. 14.86):

- **Frame Rate Display** – вычисляет и отображает фреймы оценки сигналов;
- **To Multimedia File** – записывает видеофреймы и/или отсчеты аудиосигналов в мультимединый файл;
- **To Video Display** – устанавливает видеоданные на выходное видеоустройство, видеокамеру, видеомонитор, окно вывода изображений или экран дисплея компьютера;
- **Video To Workspace** – экспортирует видеосигнал в рабочее пространство системы MATLAB;
- **Video Viewer** – представляет бинарные, полутоновые или RGB-изображения или видеопотоки;
- **Write AVI File** – записывает видеофреймы без компрессии в AVI-файлы;
- **Write Binary File** – записывает бинарные видеоданные в файлы;
- **MPlay** – просматривает видео из рабочего пространства MATLAB, файла или Simulink-сигнала.

И в этом разделе набор блоков ограничен теми блоками, которые позволяют вести запись в файлы соответствующих форматов и в рабочее пространство MATLAB.

14.6.6. Раздел Analysis & Enhancement

Раздел библиотеки по анализу и повышению качества изображений и видеопотоков представлен следующими блоками (рис. 14.88):

- **Edge Detection** – обеспечивает выделение границ фрагментов изображений, используя методы Собеля (Sobel), Превитта (Prewitt) или Робертса (Roberts);
- **Histogram Equalization** – корректирует контраст изображений, используя гистограмму распределения яркости изображения;
- **Median Filter** – осуществляет двумерную медианную фильтрацию изображений;
- **SAD** – осуществляет двумерное суммирование абсолютных разностей (sum of absolute differences – SAD).

Здесь также набор блоков ограничен – реализована лишь небольшая часть средств, которые обеспечивает в командном режиме работы пакет **Image Processing Toolbox**. Используются алгоритмы анализа, описанные для последнего пакета.

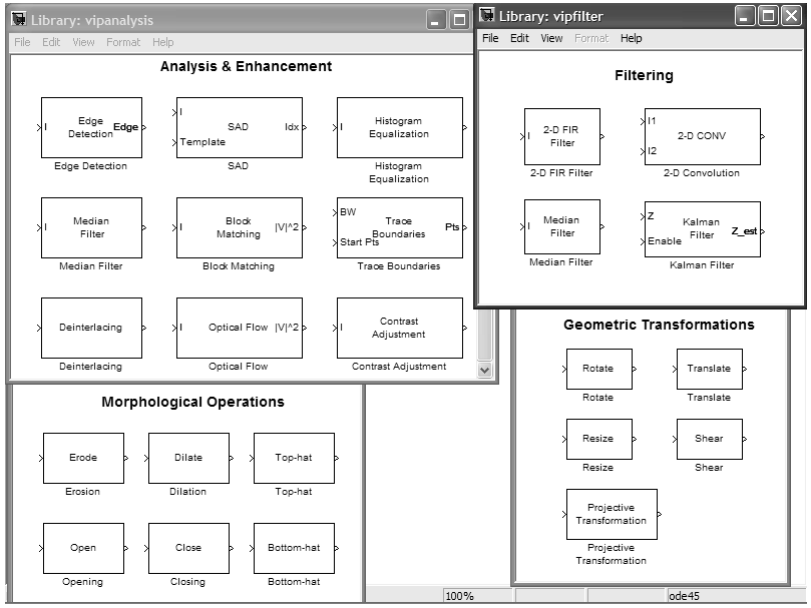


Рис. 14.88. Разделы библиотеки пакета Video and Image Processing Blockset: Analysis & Enhancement, Filtering, Morphological Operations и Geometric Transformations

14.6.7. Раздел Filtering

Фильтрация изображений часто используется для их обработки и коррекции. В разделе библиотеки **Filtering (Фильтрация)** представлены следующие три блока:

- **2-D Convolution** – вычисляет двумерную дискретную свертку двух входных матриц;
- **2-D FIR Filter** – выполняет двумерную FIR-фильтрацию входной матрицы;
- **Median Filter** – выполняет двумерную медианную фильтрацию.

Раздел фильтрации реализует минимально необходимый набор фильтров для обработки изображений. Многие современные типы фильтров, например адаптивные и на основе вейвлет-преобразований, пока не представлены.

14.6.8. Раздел геометрических преобразований Geometric Transformations

Раздел геометрических преобразований Geometric Transformations содержит ряд блоков:

- **Resize** – изменение размеров изображений;
- **Rotate** – поворот изображений на заданный угол;

- **Shear** – обеспечивает линейный сдвиг каждой строки или вектора изображения;
- **Translate** – транслирует двумерное изображение, используя вектор перемещения.

Здесь мы также имеем выборочный набор средств, подобных описанным для пакета **Image Processing Toolbox**.

14.6.9. Блоки морфологических операций – Morphological Operations

В блоке морфологических операций имеются следующие блоки:

- **Bottom-hat** – осуществляет bottom-hat-фильтрацию полутоновых или бинарных изображений;
- **Closing** – осуществляет морфологическое закрытие бинарного или intensity-изображения;
- **Dilation** – находит локальный максимум бинарного или полутонового изображения;
- **Erosion** – осуществляет локальную эрозию бинарных или intensity-изображений;
- **Label** – помечает соединяемые компоненты бинарного изображения;
- **Opening** – осуществляет морфологическое открытие бинарного или полутонового изображения;
- **Top-hat** – осуществляет top-hat-фильтрацию полутоновых или бинарных изображений.

И здесь возможности пакета **Video and Image Processing Blockset** выглядят куда более скромными, чем у **Image Processing Toolbox**.

14.6.10. Раздел Conversions

В разделе конверсии изображений (рис. 14.89) представлены следующие блоки:

- **Autothreshold** – преобразует полутоновое изображение в бинарное;
- **Chroma Resampling** – повышает или понижает цветовые компоненты изображения;
- **Color Space Conversion** – преобразует виды изображений (R'G'B', Y'CbCr и R'G'B' в to intensity);
- **Gamma Correction** – обеспечивает включение или выключение гамма-коррекции изображений и видеопотоков;
- **Image Complement** – вычисляет дополнительные значения пикселей в бинарных, полутоновых или RGB-изображениях;
- **Demosaic** – преобразует входное изображение в Bayer-формат;
- **Image Data Type Convertion** – преобразование типов данных изображений;
- **Image Data Type Conversion** – преобразование и масштабирование входных изображений в формате выходных данных.

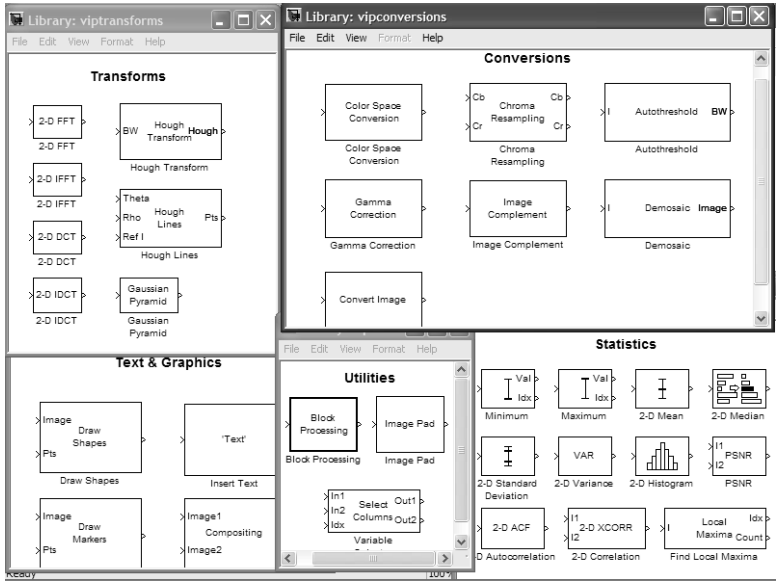


Рис. 14.89. Разделы библиотеки пакета Video and Image Processing Blockset: Conversion, Transform, Text&Graphics, Utilities и Statistics

14.6.11. Раздел Transform

В этом разделе содержатся блоки для осуществления двумерных преобразований типа 2-D FFT (быстрое преобразование Фурье) и 2-D DCT (дискретное косинусное преобразование):

- **2-D DCT** – осуществляет двумерное дискретное косинусное преобразование (DCT – discrete cosine transform);
- **2-D FFT** – осуществляет двумерное быстрое преобразование Фурье (FFT – Fast Fourier Transform);
- **2-D IDCT** – осуществляет двумерное инверсное дискретное косинусное преобразование (IDCT);
- **2-D IFFT** – осуществляет двумерное обратное быстрое преобразование Фурье (IFFT);
- **Hough Transform** – находит прямые линии в изображениях;
- **Hough Lines** – нахождение прямоугольных координат линий (rho и theta).

Это наиболее ценные блоки, обеспечивающие современные методы обработки изображений, например их сжатия, очистки от шумов и т. д. В разделе справки **References** можно найти детальные данные о выполняемых этими блоками операциях с приведением аналитических соотношений для них и детальным описанием алгоритмов.

14.6.12. Блоки статистической обработки изображений – Statistics

Раздел статистической обработки изображений содержит ряд блоков, которые обрабатывают матрицы, представляющие изображения и фреймы видеоданных. Наименование и назначение этих блоков представлены ниже.

- **2-D Autocorrelation** – вычисляет двумерную автокорреляцию входной матрицы;
- **2-D Correlation** – вычисляет двумерную кросс-корреляцию двух входных матриц;
- **2-D Histogram** – генерирует гистограмму для каждой входной матрицы;
- **2-D Maximum** – находит максимальное значение в каждой входной матрице;
- **2-D Mean** – находит среднее значение для каждой входной матрицы;
- **2-D Median** – находит медианное значение для каждой входной матрицы;
- **2-D Minimum** – находит минимальное значение каждой входной матрицы;
- **2-D Standard Deviation** – находит стандартное отклонение для каждой входной матрицы;
- **2-D Variance** – вычисляет значение варианта для каждой входной матрицы;
- **Blob Analysis** – вычисление статистических значений для выделенных областей;
- **Find Local Maxima** (Statistics) – нахождение локальных максимумов у матриц.

В разделе **Reference** справки по описываемому пакету приведены формулы, положенные в основу вычислений, реализованных данными функциями. Несмотря на их достаточно хорошую известность из курсов статистики, полезно просмотреть конкретные реализации данных формул и рекомендации по применению этих блоков.

14.6.13. Блоки раздела Text & Graphics

Этот раздел библиотеки имеет следующие блоки (рис. 14.89):

- **Compositing** – комбинирует значения пикселей двух изображений или накладывает одно изображение на другое;
- **Draw Shape** – рисует прямоугольник вокруг области интереса – region of interest (ROI);
- **Draw Markers** – отметка областей окружностей, х-меток, знаков плюс, звездочек или квадратов;
- **Draw Shapes** – отметка прямоугольников, линий, полигонов или окружностей в изображениях;
- **Insert Text** – рисует текст в изображении или видеопотоке.

14.6.14. Блоки утилит – Utilities

В состав раздела утилит входят всего три блока:

- **2-D Pad** – расширение матриц вдоль строк и/или столбцов;
- **Block Processing** – блок для создания субблоков, функции которых могут определяться пользователем;
- **Variable Selector** – селекция subset-строк и столбцов из входного сигнала.

14.7. Основные операции с изображениями и видеофайлами

14.7.1. Импорт и экспорт мультимедийных файлов

Импорт и экспорт мультимедийных файлов – наиболее часто используемые операции при работе с видеоданными. Мы уже приводили пример осуществления импорта AVI-файлов изображений (рис. 14.87). А теперь на базе созданной модели (рис. 14.90) создадим модель, которая позволит нам осуществлять экспорт таких файлов с коррекцией интенсивности зеленой и голубой компонент сигнала. Для этого добавим в модель два блока линейного преобразования сигналов **Gain** и блок записи (экспорта) изображений в AVI-файл **Write AVI File** – рис. 14.90.

Каждый блок в модели имеет свое окно настройки, с помощью которого можно задать необходимые параметры блока. Большинство параметров задаются по умолчанию, но некоторые требуют установки их пользователем. Например, для блока

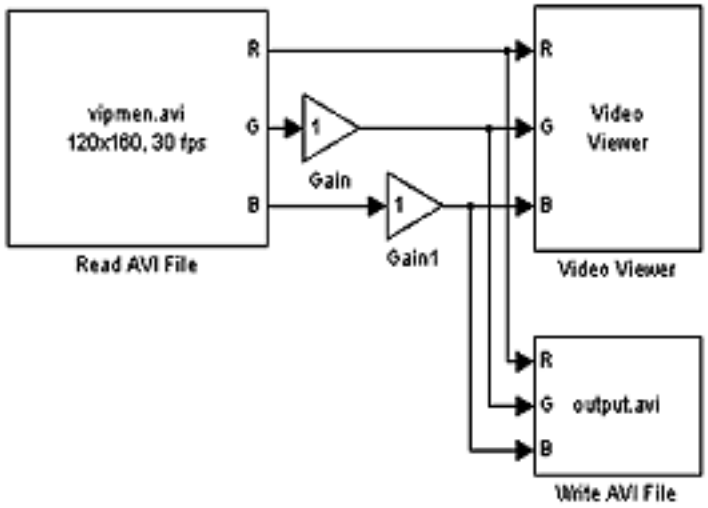


Рис. 14.90. Простая модель импорта/экспорта AVI-файлов

записи AVI-файлов надо установить имя файла (например, my_test_file.avi) и тип изображения (RGB).

В блоках **Gain** устанавливаются значения коэффициентов передачи **Gain** и тип сигнала **Same as import**. После этого можно пустить нашу модель на исполнение. Теперь, помимо просмотра выбранного AVI-файла, будет записан новый файл с именем my_test_file.avi со скорректированными уровнями зеленого и голубого цветов (если значения **Gain** оставлены равными 1, то выходной файл модели будет идентичен входному файлу). Заметим, что этот файл представлен кадрами с размером 120×160 пикселей при частоте смены кадров, равной 30 fps (30 кадров в секунду).

Вы можете поэкспериментировать с этой моделью и убедиться в том, насколько изменение параметра **Gain** одноименных с ним блоков влияет на цветовую гамму просматриваемого видеофильма. Мы, однако, не приводим вид окна просмотра видеофильма, поскольку из-за черно-белой печати рисунков в книге эти изменения слабо заметны и вид этого окна практически идентичен уже показанному – рис. 14.87.

14.7.2. Удаление шума на изображении

Передаваемые по каналам связи изображения нередко засорены шумами. Очень часто шум напоминает пищу, посыпанную солью или зернышками перчика (Salt and Pepper Noise). Такие изображения имеют характерные маленькие точки как на светлых, так и на темных участках изображения. Средствами системы MATLAB нетрудно создать такие зашумленные изображения. Например, для этого можно воспользоваться следующими командами:

```
>> I=imread(?circles.png?);
>> I=imnoise(I,?salt & pepper?, 0.02);
```

Эти команды создают в рабочем пространстве MATLAB изображение светлых окружностей на темном фоне, зашумленное точками.

Для удаления такого шума весьма эффективно применение медианного фильтра. Этот метод и реализует модель, показанная на рис. 14.91. Она вполне очевидна: блок **Image From Workspace** выделяет данные из рабочего пространства MATLAB, блок **Median Filter** очищает данные от шума, а блоки **Video Viewer** позволяют просматривать исходное и отфильтрованное изображения.

Окна вьюверов с исходным и отфильтрованным изображениями представлены на рис. 14.91 под диаграммой модели. Эффект от очистки в данном случае вполне очевиден.

14.7.3. Удаление периодического шума видеоизображений

Видеоизображения, получаемые от видеомагнитофонов, часто имеют характерный периодический шум, видимый на изображении в виде мелких клеточек или муара. Очистка видеоизображений от такого шума существенно улучшает качество изображений. Рассмотрим пример на удаление такого шума – рис. 14.92.

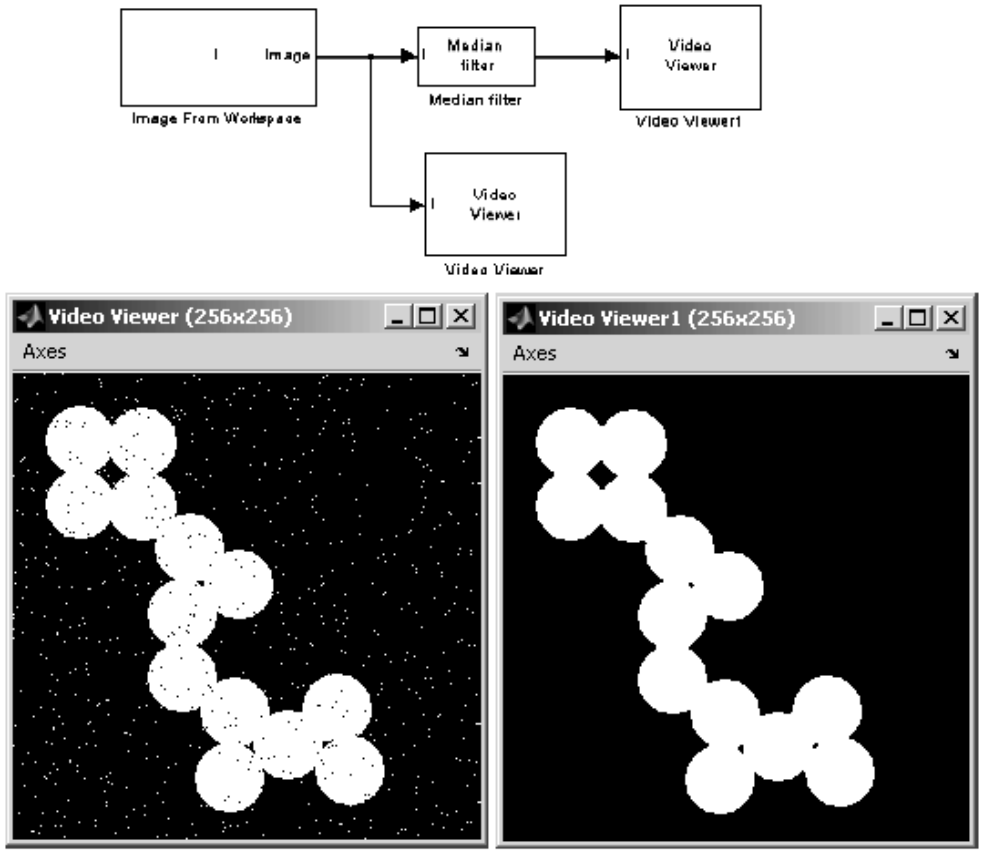


Рис. 14.91. Модель, демонстрирующая очистку изображения от шума

Эта модель обеспечивает считывание демонстрационного файла vat_video.avi, наложение на него шума (с помощью блоков **pnoise** и **Add**), очистку зашумленного сигнала с помощью фильтра **2-D FIR Filter** и представление зашумленного и очищенного сигналов с помощью блоков вьюверов.

Вьюверы модели, окна которых расположены снизу рис. 14.92, позволяют наблюдать исходное изображение (показан стоп-кадр), зашумленное изображение и очищенное изображение. Видно, что очистка не является идеальной. Она более-менее хороша в центре, но по краям наблюдаются искажения изображения, называемые артефактами.

14.7.4. Создание панорамного изображения

Другой интересный пример – создание панорамного изображения из некоторого исходного изображения. Диаграмма этого примера представлена на рис. 14.93 сверху. Создание панорамы сводится к сканированию исходного изображения от

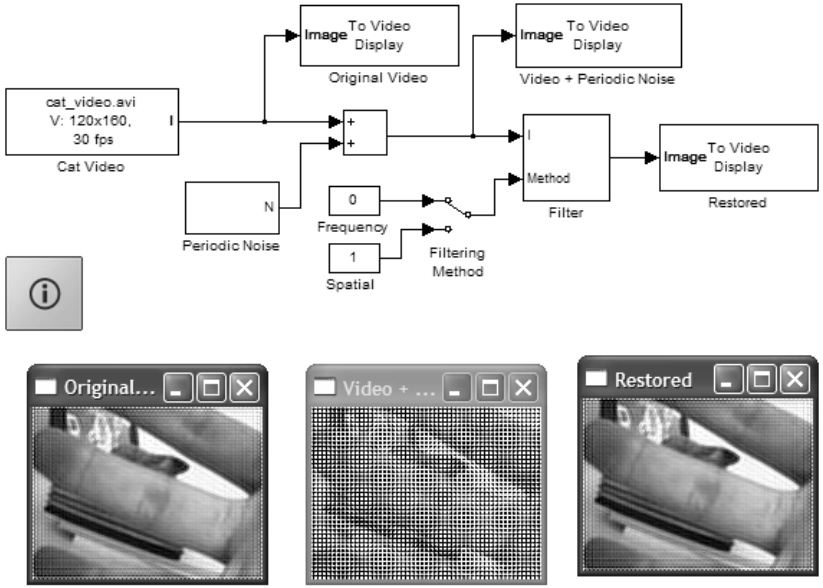


Рис. 14.92. Модель, демонстрирующая очистку видеоизображения от периодического шума

видеоисточника Video Source, которое контролируется и отображается блоком **Video Viewer (Input)**. Результат сканирования можно наблюдать в виде более широкой картинки, которая разворачивается на экране, – на рис. 14.93 снизу показан лишь заключительный кадр этого процесса.

Рекомендуется просмотреть субблоки этой диаграммы. Они позволят судить о сложности реализации данной операции.

14.7.5. Построение динамической картинки внутри другой динамической картинки

В разделе демонстрационных примеров **Miscellaneous** появились три новых интересных примера. Один из них демонстрирует создание динамической картинки внутри другой динамической картинки. Диаграмма этого примера представлена на рис. 14.94.

Эта диаграмма предельно проста. Основным является блок **PIP Display** (PIP – от слова Picture in Picture). Видеоизображения основной и дополнительной картинок поступают с видеоканалов (субблок **Chanels**) и представляют собой видеофильм, снятый из движущегося автомобиля, – на рис. 14.94 виден фрагмент этого фильма, показывающий дорогу с объектами на ней и около нее. Блок контроля **Control** служит для задания параметров окна просмотра.

Внизу диаграммы (рис. 14.94) имеются два переключателя, которые задают параметры **slide** и **fade** (увядание) изображения картинки в картинке. При пуске

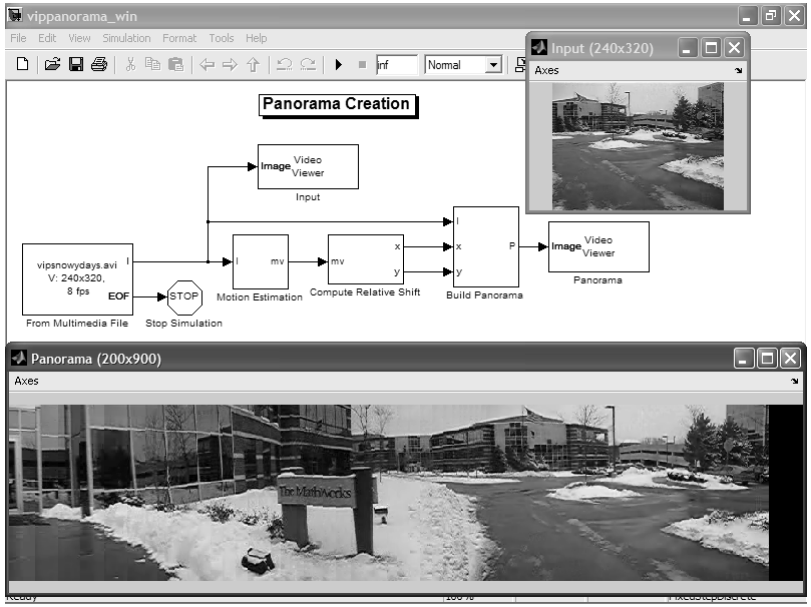


Рис. 14.93. Диаграмма модели создания панорамного изображения, исходное изображение и панорамное изображение (снизу)

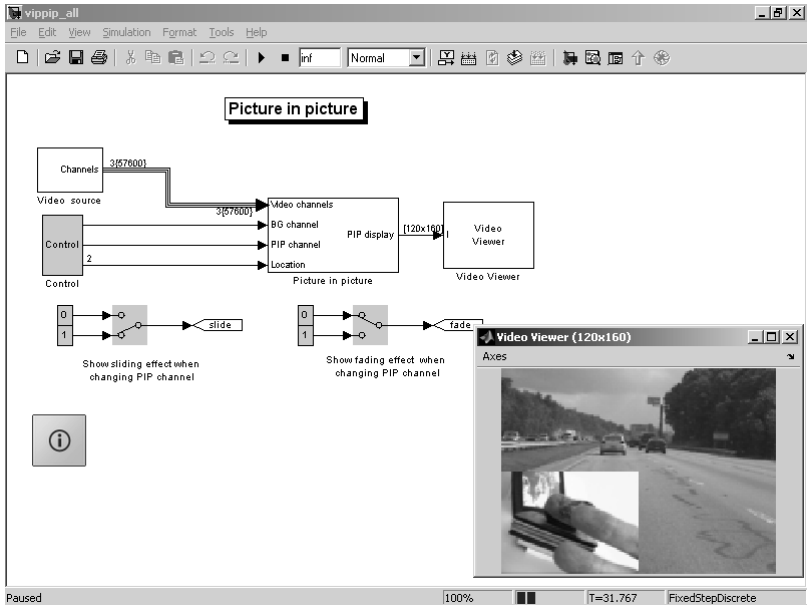


Рис. 14.94. Пример представления картинки в картинке

можно наблюдать появление картинки в картинке и ее перемещение в конечное положение – левый нижний угол. На рис. 14.94 представлен лишь один кадр просмотра, на самом деле как сама основная картинка, так и картинка в картинке представляют собой кадры видеофильмов. Таким образом моделируется эффект картинки в картинке, широко используемый в современных телевизионных приемниках и видеомониторах компьютеров.

14.7.6. Вращение изображения

Непрерывное вращение изображения иллюстрирует пример, диаграмма которого представлена на рис. 14.95. Эта диаграмма настолько проста, что не нуждается в особых комментариях. Вращение дает блок **Rotate Angle**, обеспечивающий пересчет координат каждой точки изображения с помощью матрицы вращения.

В пакете **Video and Image Processing Blockset** имеется и специальный блок **Rotate** для осуществления поворота изображений, представленных матрицами. Блок **Rotate** имеет два входа. Первый верхний вход I служит для подачи на него данных матрицы A, а второй Angle – для подачи сигнала, задающего угол поворота изображения angle. Для повышения качества повернутого изображения этот блок использует интерполяцию 2D-данных. Возможны следующие виды интерполяции:

- **Nearest Neighbor** – интерполяция по ближайшим точкам;
- **Bilinear** – билинейная интерполяция;
- **Bicubic** – бикубическая интерполяция.

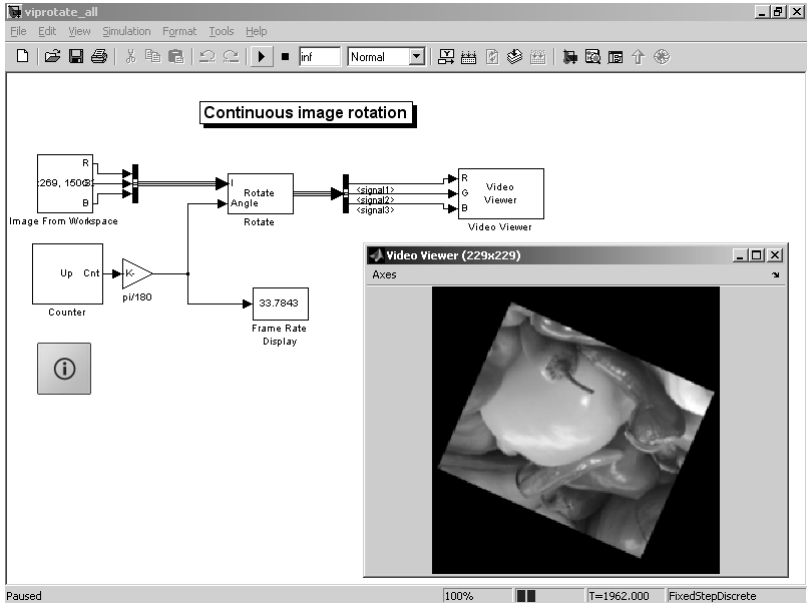


Рис. 14.95. Диаграмма примера на непрерывное вращение изображения

14.7.7. Морфологическое открытие и пересчет объектов изображения

Пересчет объектов в изображении иллюстрирует эффективность выполнения морфологических операций. Рассмотрим применение этой операции на примере изображения из файла testpat1.png. Для загрузки его в рабочее пространство MATLAB исполним команду

```
>> I=imread(?testpat1.png?)
```

Это тестовое изображение представляет собой черный круг с 24 светлыми сегментами. Оно представлено матрицей I размера 256×256 элементов в виде 8-битовых чисел без знака.

Модель, решающая нашу задачу, представлена на рис. 14.96. Блок **Opening** осуществляет морфологическое открытие изображения с учетом близости пикселей фрагментов изображения, а блок **Labels** – подсчет числа светлых объектов. Назначение остальных блоков модели вполне очевидно.

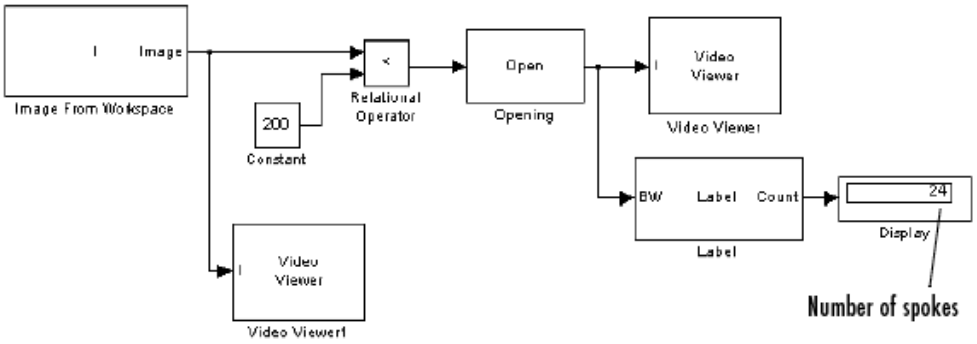


Рис. 14.96. Диаграмма модели, демонстрирующая открытие и пересчет объектов изображения

В окне параметров блока **Opening** надо указать необходимые значения двух параметров блока, отражающие характер близости объектов. В блоке **Labels** установить три параметра: число соединений, тип выхода и тип выходных данных. Вьюверы служат для просмотра исходного и преобразованного. Преобразованное изображение содержит четко разделенные объекты, что необходимо для их подсчета блоком **Labels**. Число объектов на диаграмме модели показывает блок **Display** – на рис. 14.96 показания дисплея выделены надписью Number of spokes. Нужны следующие установки параметров моделирования (раздел Solver): Stop time=0, Type=fixed-step и Solver=discrete.

14.7.8. Улучшение четкости выделенной части изображения

Диаграмма модели, показанная на рис. 14.97, демонстрирует сразу несколько интересных возможностей пакета **Video and Image Processing Blockset**: считывание AVI-файла и его отображение, задание и построение прямоугольника, выделяющего часть изображения и текстовой надписи в нем, и, наконец, фокусировку выделенной части изображения.

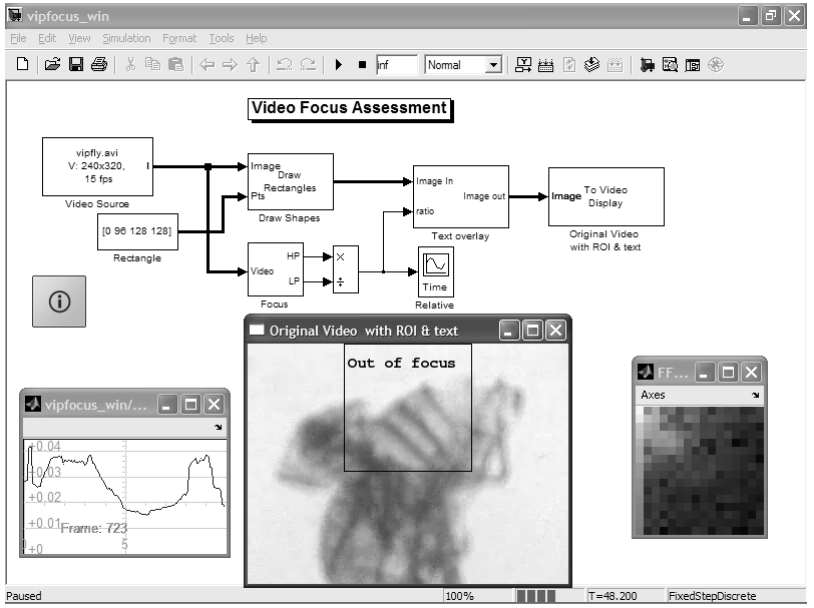


Рис. 14.97. Диаграмма модели, демонстрирующая выделение части изображения и его фокусировку

14.7.9. Нахождение и выделение кромок у объектов изображений

Нахождение и выделение кромок (границ) у объектов изображений также являются довольно частой задачей при обработке изображений и создании различных художественных эффектов. В справке по пакету есть примеры на выделение кромок как неподвижных изображений (бактерий), так и подвижных, представлен-

ных AVI-файлами. На рис. 14.98 показана диаграмма модели, обеспечивающая выделение границ различных объектов (включая контуры людей) в кадрах видеофильма.

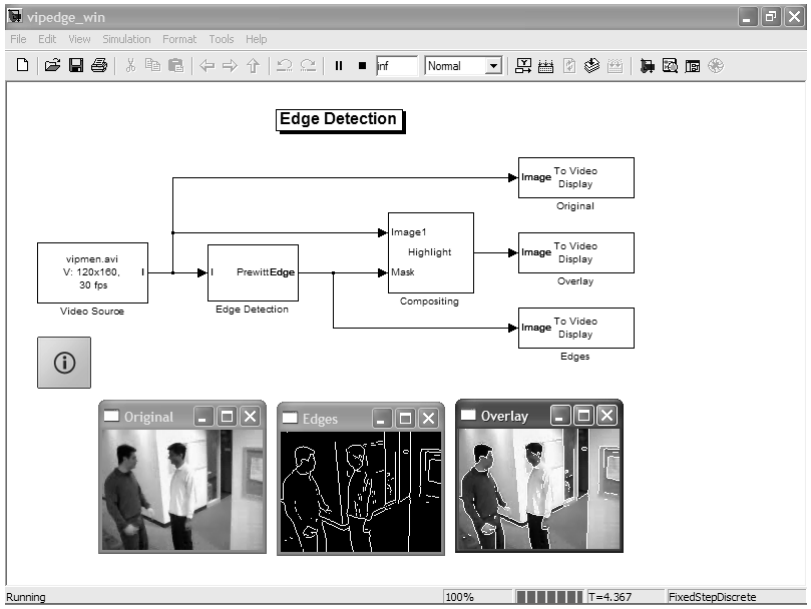


Рис. 14.98. Диаграмма модели выделения границ кадров видеофильма

14.7.10. Стабилизация перемещаемого изображения

При видеосъемке часто возникает необходимость в стабилизации быстро и неравномерно перемещающегося изображения – например, автомобиля, едущего по дороге. Для этого служат различные электронные и программные стабилизаторы положения графических объектов. Смоделировать работу таких устройств позволяет модель, представленная на рис. 14.99. Ее можно открыть, используя команду vipstabilize в командном режиме MATLAB. В модели используется блок **Translation Stabilization**.

Работа таких устройств (и моделей) основана на захвате объекта изображения, например автомобиля – рис. 14.100. После этого перемещение объекта в «поле зрения» видеокамеры практически исключается, и объект будет находиться в заданном месте – рис. 14.101.

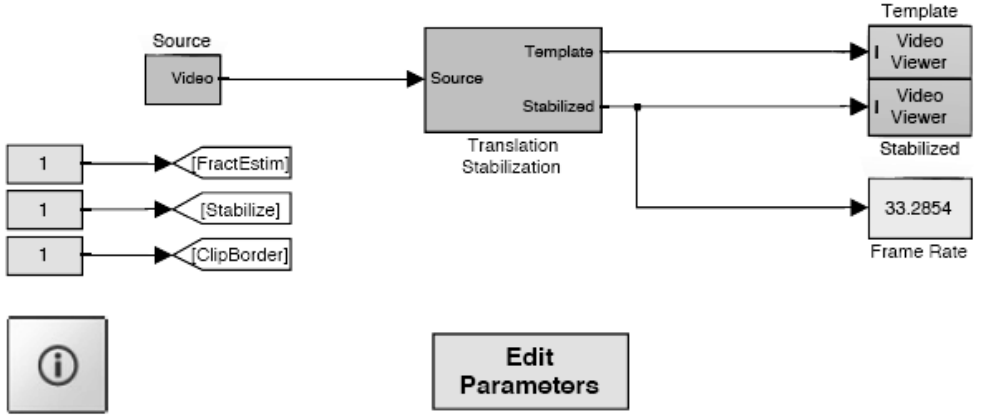


Рис. 14.99. Модель стабилизации положения объекта

Рис. 14.100. Захват графического объекта – автомобиля

Рис. 14.101. Кадр перемещения автомобиля с сохранением его в центре кадра

14.7.11. Прослеживание движущихся автомобилей

Одной из актуальных задач автоматизации движения на автомагистралях является прослеживание движущихся автомобилей. В справке по пакету **Video and Image Processing Blockset** есть интересные примеры на решение этой задачи. На рис. 14.102 показана диаграмма модели, которая из видеофайла оригинала выделяет движущиеся автомобили, используя преобразование R"G"B" составляющих сигнала в их интенсивность и блок **Optical Flow** для отделения их от изображения дороги.

Другая модель (рис. 4.103) использует для прослеживания движения автомобилей оценку фона (дороги и окружающих ее предметов) с помощью блока **Background Estimator**.

14.7.12. Сегментация по цвету и ячеек

Сегментация означает разделение или выделение сегментов изображения. На рис. 14.104 показана диаграмма модели цветной сегментации динамического изображения – лица человека. Диаграмма очень проста и основана на использовании блока.

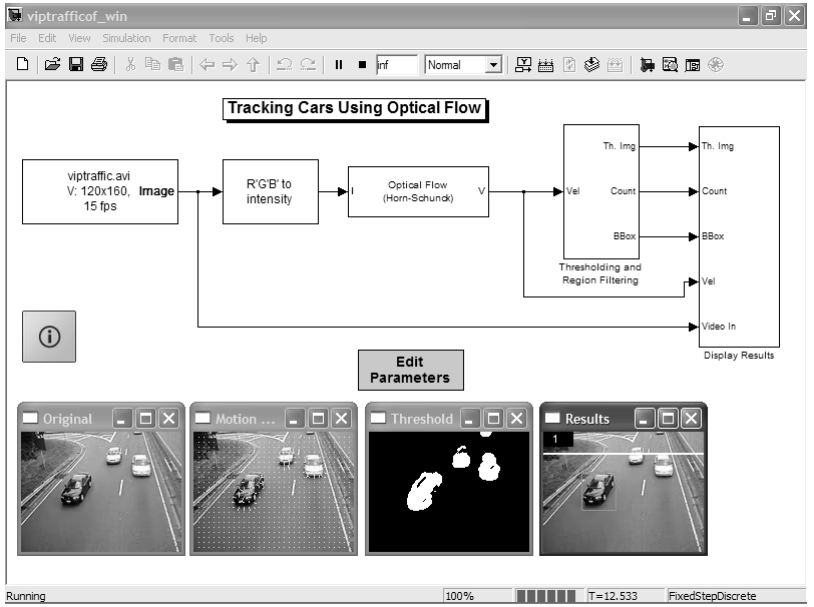


*Рис. 14.102. Диаграмма модели прослеживания за движущимся автомобилем с использованием блока **Optical Flow***

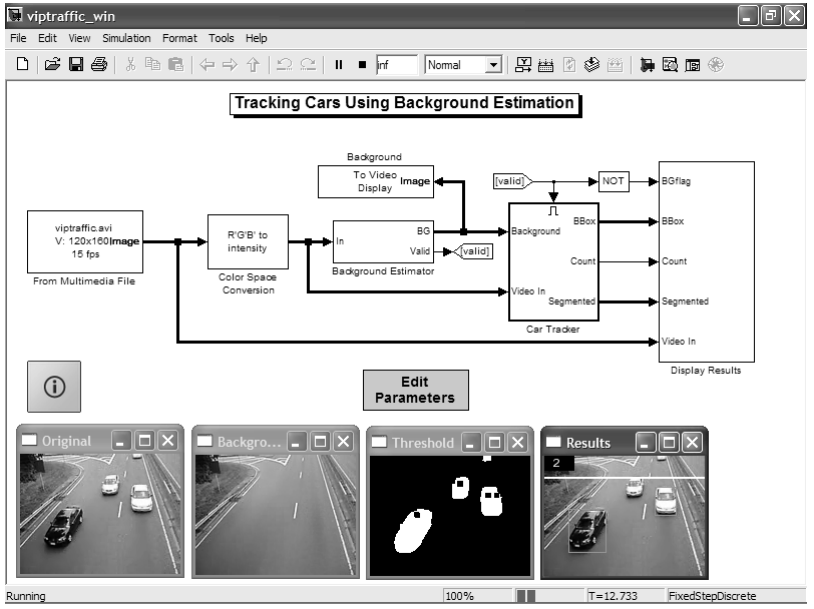


Рис. 14.103. Диаграмма модели прослеживания за движущимся автомобилем с использованием блока Background Estimator

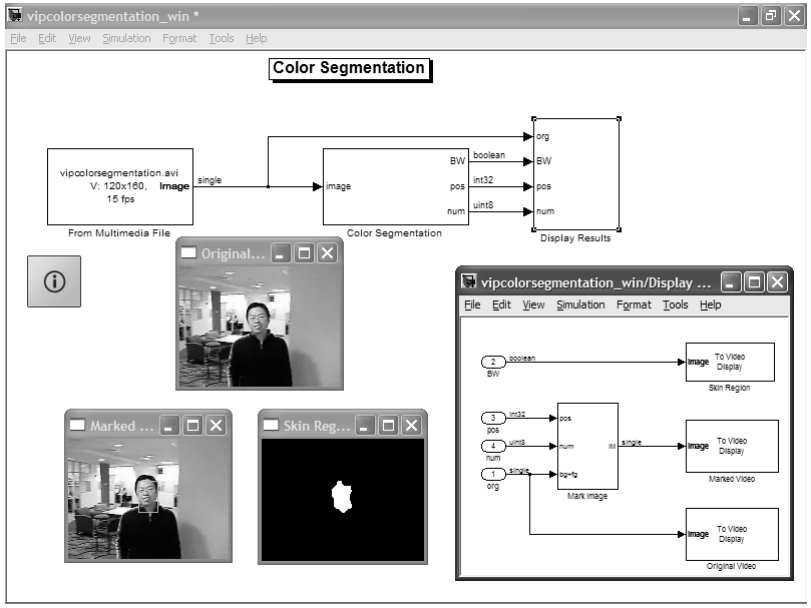


Рис. 14.104. Диаграмма модели сегментации по цвету

Еще один красочный пример сегментации ячеек изображения представляет диаграмма модели, показанная на рис. 14.105. Здесь для сегментации используются блоки **Segment Cells** и **Blob Analisys**.

14.7.13. Сжатие изображения

Сжатие изображений в наше время получило широкое распространение. Наиболее часто используются алгоритмы блочного сжатия, основанные на применении косинусных преобразований. Простой и чисто учебный пример реализации сжатия изображений представлен ниже.

Сжатию подлежит изображение фотографа из файла cameraman.tif. Как обычно, его надо загрузить в рабочее пространство MATLAB, исполнив команду

```
>> I=imread(?cameraman.tif?)
```

Модель, демонстрирующая средства сжатия изображения, представлена на рис. 14.106. Она упрощена за счет применения двух подмоделей (субблоков) на основе блока **Block Processing**.

Как принято при работе с пакетом Simulink, для просмотра подмодели достаточно щелкнуть по ней мышью. Для первой подмодели Block Processing (без номера) это открывает окно подмодели, представленное на рис. 14.107. Подмодель очень проста – помимо портов ввода In1 и вывода Out, она содержит блок двумерного прямого дискретного косинусного преобразования **2-D DCT и Selector**. Они и обеспечивают сжатие сигналов изображения.

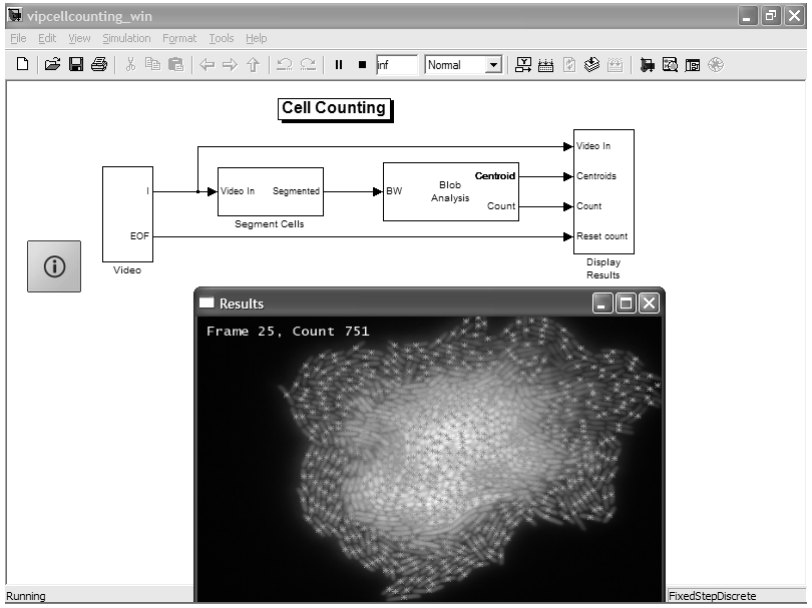


Рис. 14.105. Диаграмма модели сегментации ячеек

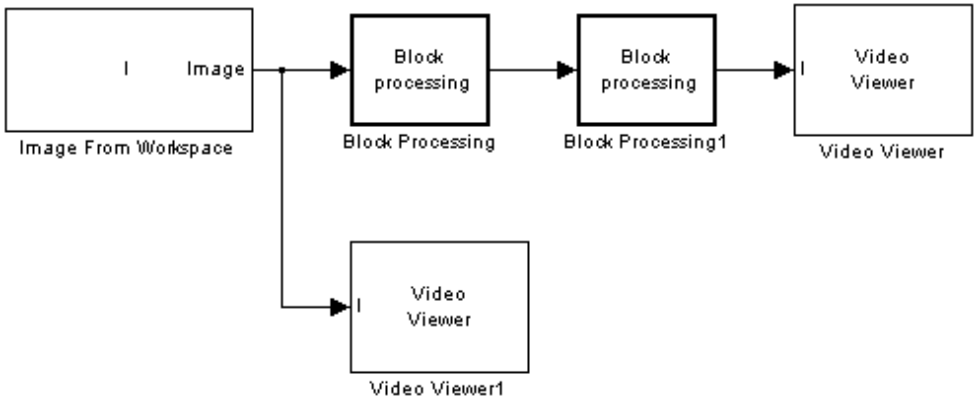


Рис. 14.106. Модель, демонстрирующая сжатие и восстановление изображения

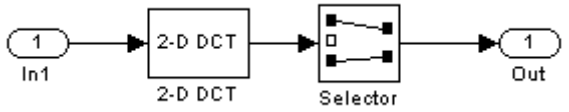


Рис. 14.107. Подмодель сжатия сигналов изображения

Блок **2-D DCT** реализует стандартную процедуру разбивки изображения на блоки заданного размера 8×8 и алгоритм прямого дискретного косинусного преобразования. Блок обеспечивает преобразование матрицы изображения в последовательность косинусных составляющих, которую можно затем превратить опять в изображение.

Полное прямое и обратное дискретное косинусное преобразование полностью восстанавливает сигнал и по существу в таком виде бессмысленно. Однако оно может использоваться для сжатия сигнала путем удаления части его составляющих, незаметных на глаз. Для этого в подмодель сжатия изображения включен блок **Selector**. Этот блок и ограничивает поток данных изображения.

Для того чтобы наблюдать сжатое изображение, служит блок Block Processing 1, обеспечивающий расширение потока данных и обратное дискретное косинусное преобразование. Диаграмма этой подмодели представлена на рис. 14.108.

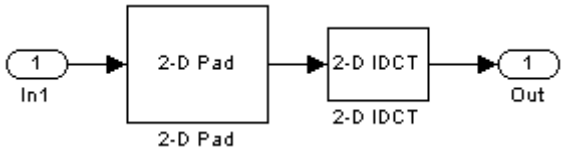


Рис. 14.108. Подмодель расширения изображения

На рис. 14.109 и 14.110 показаны окна вьюверов. Первое окно **Video Viewer 1** показывает исходное изображение, а второе **Video Viewer** – сжатое изображение, восстановленное до первоначального размера. Можно заметить (в рисунках книги с трудом), что эти изображения немного различаются – качество сжатого изображения немного хуже, чем исходного. Это вполне закономерно, поскольку данный метод сжатия построен на основе алгоритма удаления мелких деталей изображения.

Рис. 14.109. Окно вьювера с исходным изображением фотографа

Рис. 14.110. Окно вьювера со сжатым изображением фотографа

14.7.14. Проекция изображения на вращающийся кубик

Последний из примеров пакета **Video and Image Processing Blockset** в данном разделе демонстрирует проекцию изображения на вращающийся кубик с зеркальными гранями – рис. 14.111.

Приведенные в этом разделе демонстрационные примеры показывают, что пакет расширения **Video and Image Processing Blockset** превратился в мощное средство работы с видеопотоками и видеофайлами. Он существенно дополняет пакет расширения **Image Processing Toolbox**, который предоставляет обширные средства обработки статических изображений.

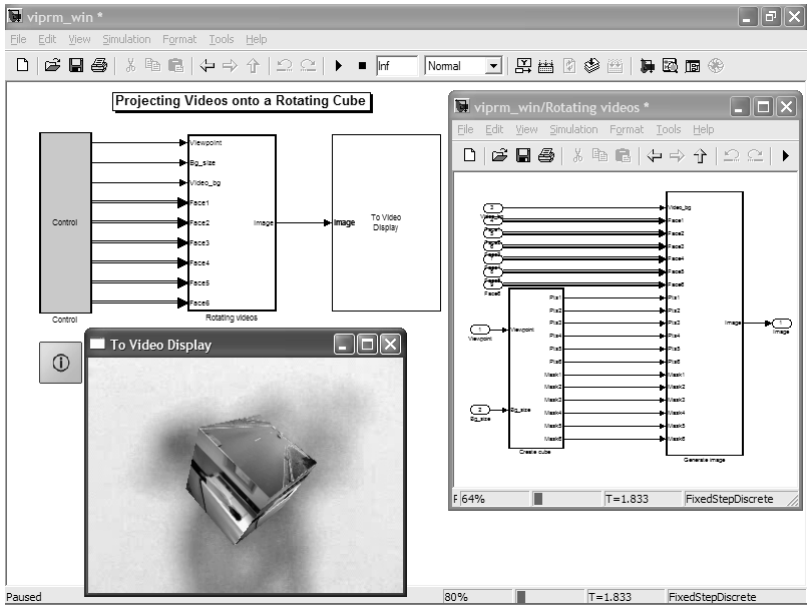


Рис. 14.111. Диаграмма модели проектирования изображения на вращающийся кубик

Список литературы

1. Дьяконов В. П. Компьютерная математика. Теория и практика. – М.: Нолидж, 2000.
2. Гантмахер Ф. Теория матриц. – М.: Наука, Физматлит, 1988.
3. Фадеев А. К., Фадеева В. Н. Вычислительные методы линейной алгебры. 3-е изд., стереотипное. – СПб.: Лань, 2002.
4. Дьяконов В. П. Справочник по применению системы PC MATLAB. – М.: Наука, Физматлит, 1993.
5. Дьяконов В. П., Абраменкова И. В. MATLAB 5.0/5.3. Система символьной математики. – М.: Нолидж, 1999.
6. Дьяконов В. П., Абраменкова И. В., Круглов В. В. MATLAB 5 с пакетами расширений. – М.: Нолидж, 2001.
7. Дьяконов В.П. MATLAB: Учебный курс. – СПб.: ПИТЕР, 2001.
8. Дьяконов В.П. MATLAB 6: Учебный курс. – СПб.: ПИТЕР, 2001.
9. Дьяконов В. П. Simulink 4: Специальный справочник. – СПб.: ПИТЕР, 2002.
10. Дьяконов В. П., Круглов В. В. Математические пакеты расширения MATLAB: Специальный справочник. – СПб.: ПИТЕР, 2001.
11. Дьяконов В. П., Круглов В. В. MATLAB. Анализ, идентификация и моделирование систем: Специальный справочник. – СПб.: ПИТЕР, 2002.
12. Дьяконов В. П., Абраменкова И. В. MATLAB. Обработка сигналов и изображений: Специальный справочник. – СПб.: ПИТЕР, 2002.
13. Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5 в. Основы применения: Полное руководство пользователя. – М.: Солон-Р, 2002.
14. Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5 в в математике и моделировании: Полное руководство пользователя. – М.: Солон-Р, 2002.
15. Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5 в. Обработка сигналов и изображения: Полное руководство пользователя. – М.: Солон-Р, 2004.
16. Дьяконов В. П. MATLAB 6.5 SP1/7 + Simulink 5/6 в. Основы применения. – М.: Солон-Р, 2005.
17. Дьяконов В. П. MATLAB 6.5 SP1/7 + Simulink 5/6 в в математике и моделировании. – М.: Солон-Р, 2005.
18. Дьяконов В. П. MATLAB 6.5 SP1/7 + Simulink 5/6 в. Обработка сигналов и проектирование фильтров. – М.: Солон-Р, 2005.
19. Дьяконов В. П. MATLAB 6.5 SP1/7 + Simulink 5/6 в. Работа с изображениями и видеопотоками. – М.: Солон-Р, 2005.
20. Дьяконов В. П., Круглов В. В. MATLAB 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6 в. Инструменты искуственного интеллекта и биоинформатики. – М.: Солон-Пресс, 2006.
21. Дьяконов В. П. VisSim+Mathcad+MATLAB. Визуальное математическое моделирование. – М.: Солон-Пресс, 2004.

22. Потемкин В. Г. MATLAB 5 для студентов. – М.: ДИАЛОГ-МИФИ, 1998.
23. Потемкин В. Г. Система инженерных и научных расчетов MATLAB 5.*. – В 2 т. – М.: ДИАЛОГ-МИФИ, 1999.
24. Потемкин В. Г. Инструментальные средства MATLAB 5.*. – М.: ДИАЛОГ-МИФИ, 2000.
25. Потемкин В. Г. Вычисления в среде MATLAB. – М.: ДИАЛОГ-МИФИ, 2004.
26. Лазарев Ю. Ф. MATLAB 5.* (серия «Библиотека студента»). – Киев: Издательская группа BHV, 2000.
27. Ануфриев И. MATLAB 5.3/6.*: Самоучитель. – СПб.: БХВ-Петербург, 2002.
28. Ануфриев И. Е., Смирнов А. Б., Смирнова Е. Н. MATLAB 7. – СПб.: БХВ-Петербург, 2005.
29. Кетков Ю., Кетков А., Шульц М. MATLAB 6.*: программирование численных методов. – СПб.: БХВ-Петербург, 2004.
30. Кетков Ю. Л., Кетков А. Ю., Шульц М. М. MATLAB 7. Программирование, численные методы. – СПб.: БХВ-Петербург, 2005.
31. Чен К., Джиблин П., Ирвинг А. MATLAB в математических исследованиях. – М.: Мир, 2001.
32. Гультяев А. Визуальное моделирование в среде MATLAB: Учебный курс. – СПб.: Питер, 2000.
33. Черных И. В. SIMULINK. Среда создания инженерных приложений. – М.: ДИАЛОГ-МИФИ, 2004.
34. Лазарев Ю. Моделирование процессов и систем в MATLAB: Учебный курс. – СПб.: Питер; Киев: Изд. группа BHV, 2005.
35. Медведев В. С., Потемкин В. Г. Control System Toolbox. MATLAB 5 для студентов. – М.: ДИАЛОГ-МИФИ, 2004.
36. Рудаков П. И., Сафонов В. И. Обработка сигналов и изображений. MATLAB 5.* / Под общ. ред. В. Г. Потемкина. – М.: ДИАЛОГ-МИФИ, 2000.
37. Лавров К. Н., Цыплякова Т. П. Финансовая аналитика. MATLAB 6 / Под общ. ред. В. Г. Потемкина. – М.: Диалог-МИФИ, 2001.
38. Мартынов Н.Н., Иванов А. П. MATLAB 5.*. Вычисления, визуализация, программирование. – М.: КУДИЦ-ОБРАЗ, 2000.
39. Герман-Галкин С. Г. Компьютерное моделирование полупроводниковых систем в MATLAB 6.0. – СПб.: Корона, 2001.
40. Говорухин В., Цибулин В. Компьютер в математических исследованиях: Учебный курс. – СПб.: ПИТЕР, 2001.
41. Дьяконов В. П. Вейвлеты. От теории к практике. 2-е изд., доп. и перераб. – М.: СОЛОН-Пресс, 2004.
42. Медведев В. С., Потемкин В. Г. Нейронные сети. MATLAB 6. – М.: ДИАЛОГ-МИФИ, 2002.
43. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ-Петербург, 2003.
44. MATLAB. The Language of Technical Computing. Getting Started with MATLAB. The Math Works, Inc. USA, 2000.

45. MATLAB. The Language of Technical Computing. Using MATLAB. The Math Works, Inc. USA, 2000.
46. MATLAB. The Language of Technical Computing. Using MATLAB Graphics. The Math Works, Inc. USA, 2000.
47. MATLAB. The Language of Technical Computing. External Interfaces. The Math Works, Inc. USA, 2000.
48. Simulink. Model-Based and System-Based Design. Using Simulink. The Math Works, Inc. USA, 2002.
49. Numerical Computing with MATLAB (text book) The Math Works, Inc. (www.mathworks.com/moler).
50. Математический энциклопедический словарь / Под ред. Ю. В. Прохорова. – М.: Советская энциклопедия, 1988.
51. Бронштейн И. Н., Семендяев К. А. Справочник по математике для инженеров и учащихся вузов. – М.: Наука, Физматлит, 1980.
52. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров. – М.: Наука, 1973.
53. Справочник по специальным функциям с формулами, графиками и математическими таблицами / Под ред. М. Абрамовица и И. Стиган. – М.: Наука, Физматлит, 1979.
54. Ильин В. А., Позняк Э. Г. Основы математического анализа. В 2 ч. 6-е изд. – М.: Физматлит, 2001.
55. Ильин В. А., Позняк Э. Г. Линейная алгебра. – М.: Физматлит, 2001.
56. Бабенко К. И. Основы численного анализа. – М.: Наука, Физматлит, 1986.
57. Марчук Г. И. Методы вычислительной математики. – М.: Наука, Физматлит, 1989.
58. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. – М.: Наука, Физматлит, 1987.
59. Трауб Дж. Итерационные методы решения уравнений. – М.: Мир, 1985.
60. Дэннис Дж., Шнабель Р. Численные методы безусловной оптимизации и решения нелинейных уравнений / Пер. с англ., под ред. Ю. Г. Евтушенко. – М.: Мир, 1988.
61. Иванов В. В. Методы вычислений на ЭВМ: Справочное пособие. Киев: Наукова думка, 1986.
62. Аоки М. Введение в методы оптимизации. – М.: Наука, 1977.
63. Банди Б. Методы оптимизации: Вводный курс. – М.: Радио и связь, 1988.
64. Шредер М. Фракталы, хаос, степенные законы. Миниатюры из бесконечного ряда. – Ижевск: НИЦ «Регулярная и хаотическая механика», 2001.
65. Роджерс Д., Адамс Дж. Математические основы машинной графики. – М.: Мир, 2001.
66. Новые информационные технологии: Учебное пособие / Под ред. В. П. Дьяконова. – М.: СОЛОН-Пресс, 2005.
67. Дьяконов В. П. Энциклопедия Mathcad 2001i и 11. – М.: СОЛОН-Пресс, 2004.
68. Дьяконов В. П. Maple 9 в математике, физике и образовании. – М.: СОЛОН-Пресс, 2004.

69. Дьяконов В. П. Internet. Настольная книга пользователя. 5-е изд., перераб. и доп. – М.: СОЛОН-Пресс, 2005.
70. Самарский А. А., Михайлов А. П. Математическое моделирование: Идеи. Методы. Примеры. 2 изд., испр. – М.: Физматлит, 2001.
71. Бенькович Е. С., Колесов Ю. Б., Сениченков Ю. Б. Практическое моделирование динамических систем. – СПб.: БХВ-Петербург, 2002.
72. Максимей И. В. Имитационное моделирование на ЭВМ. – М.: Радио и связь, 1988.
73. Шеннон Р. Имитационное моделирование систем. Искусство и наука. – М.: Мир, 1978.
74. Семененко М. Г. Введение в математическое моделирование. – М.: Солон-Р, 2002.
75. Математическое моделирование / Под ред. Дж. Эндрюса, Р. Мак-Лоуна; пер. с англ. – М.: Мир, 1979.
76. Введение в математическое моделирование: Учебное пособие / В. Н. Ашихмин и др.; под ред. П. В. Трусова. – М.: Интермет Инжиниринг, 2000.
77. Цисарь И. Ф., Нейман В. Г. Компьютерное моделирование экономики. – М.: Диалог-МИФИ, 2002.
78. Шелобаев С. И. Математические методы и модели в экономике, финансах, бизнесе: Учебное пособие для вузов. – М.: ЮНИТИ-ДАНА, 2001.
79. Экономико-математические методы и прикладные модели: Учебное пособие для вузов / В. В. Федосеев, А. Н. Гармаш, Д. М. Дайтибегов и др.; под ред. В. В. Федосеева. – М.: ЮНИТИ, 2001.
80. Прицкер А. Введение в имитационное моделирование и язык СЛАМ II. – М.: Мир, 1987.

Предметный указатель

ИД ДМК-пресс

Москва, 1-й Силикатный пр-д, д.14
тел/факс:(495) 259-86-74 e-mail: sales@easy-read.ru
www.easy-read.ru

Издательский дом «ДМК-пресс» предлагает для продажи в сети ваших магазинов **универсальный держатель EASY-READ™ –приспособление, которое помогает читать книги и журналы в любом месте: дома, на работе, на пляже, в походе**. Иными словами **–** везде, где угодно.

Держатель устойчив на любой поверхности. Уникальным достоинством держателя является возможность расположения его не только на столе, стене, полке, но и на **мониторе компьютера.**

Универсальный держатель EASY-READ™:

- **Не имеет** аналогов в мире
- **Чрезвычайно легкий –** всего 85 граммов
- **Компактный –** в сложенном виде занимает место менее дамского зонтика
- **Изготовлен** из современных экологически чистых материалов
- **Пригоден** для документов любого размера, вплоть до формата А3
- **Предназначен** для оснащения школ, офисов, библиотек, интернет-кафе
- **Невысокая цена**, изящный чехол
- **Удобная и полезная** вещь для ежедневного использования.
- **Прекрасный подарок** любому ценителю всего необычного.

Условия сотрудничества:

- Каждой торговой точке розничной сети наша компания предоставляет красочную и эффектную витрину-стенд, привлекающую внимание покупателей, что даст возможность самостоятельно ознакомиться с достоинствами держателей **EASY-READ™**
- Адреса розничных магазинов будут размещены на сайте нашей компании и войдут в список точек, в которых можно приобрести держатели **EASY-READ™**.

Подробнее с данным продуктом можно ознакомиться на сайте www.easy-read.ru.

Приобрести оптом и в розницу держатель можно приобрести в **компании «Альянс-книга»** www.alians-kniga.ru, **тел. (495) 258-91-94, 258-91-95**

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «АЛЬЯНС-КНИГА» наложенным платежом, выслав открытку или письмо по почтовому адресу: **123242, Москва, а/я 20** или по электронному адресу: **orders@alians-kniga.ru**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в Internet-магазине: **www.alians-kniga.ru**.

Оптовые закупки: тел. **(495) 258-91-94, 258-91-95**; электронный адрес **books@alians-kniga.ru**.

Дьяконов Владимир Павлович

Simulink 5/6/7

Самоучитель

Главный редактор *Мовчан Д. А.*
dm@dmk-press.ru
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Подписано в печать 04.02.2008. Формат 70×100 $^{1}/_{16}$.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 73.5. Тираж 1500 экз.
№
Издательство ДМК Пресс
Web-сайт издательства: www.dmk-press.ru
Internet-магазин: www.abook.ru