Пусть эта книга принесет Вам удачу!

Владимир Дьяконов

VisSim+Mathcad+MATLAB

ВИЗУАЛЬНОЕ МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

Основы моделирования
Описание библиотеки блоков
Применение VisSim
Пакеты расширения системы
300 практических примеров

ПОЛНОЕ
руководство пользователя

Серия «Полное руководство пользователя»

В. П. Дьяконов

VisSim+Mathcad+MATLAB

Визуальное математическое моделирование

Москва
СОЛОН-Пресс
2004

УДК 621.396.218
ББК 32.884.1
Д93

В. П. Дьяконов

Д93 VisSim+Mathcad+MATLAB. Визуальное математическое моделирование. — М.: СОЛОН-Пресс, 2004. — 384 с.: ил. — (Серия «Полное руководство пользователя»).

ISBN 5-98003-130-8

Практическое руководство по новой перспективной системе визуального блочного математического моделирования VisSim 3/4.5/5, обеспечивающей интеграцию с массовыми системами компьютерной математики Mathcad и MATLAB. Даны основы математического моделирования, полное описание библиотеки блоков и применение VisSim для обработки, включая фильтрацию, сигналов, моделирования и анализа различных систем и устройств. Детально рассмотрены пакеты расширения системы. Дано свыше 300 наглядных примеров применения VisSim — оригинальных, демонстрационных и с использованием средств интегрированных с VisSim других программных продуктов.

Для инженеров, научных работников, преподавателей и студентов университетов и вузов.

УДК 621.396.218
ББК 32.884.1

ISBN 5-98003-130-8

Предисловие

Математическое моделирование находит самое широкое применение в науке, технике, экономике и образовании [1—12]. Первое время оно было привилегией специалистов, имеющих доступ к большим ЭВМ. Уже в конце 70-х гг. прошлого столетия появились программируемые микрокалькуляторы и микроЭВМ, позволившие индивидуальным пользователям (например, студентам и инженерам) решать простые задачи математического моделирования [13—16]. При этом программы моделирования составлялись самими пользователями или для этого использовались специальные языки моделирования, например СЛАМ II [12]. Подготовка систем уравнений состояния для моделей и их модификация при изменении моделей были трудоемким и сложным делом.

Появление в начале 80-х гг. персональных компьютеров (ПК) создало реальные условия для реализации на них уже достаточно серьезных специализированных систем математического моделирования, в которых была полностью автоматизирована подготовка как моделей, так и их уравнений состояния. Как правило, это базировалось на матричных методах и графах. Например, в электронике известность получил целый класс систем схемотехнического моделирования PSpice, Design Lab, MicroCAP, Electronic Workbench и др. [17—19]. Появились, правда, менее известные программы моделирования физических (в том числе механических), химических и иных устройств и систем.

Параллельно с разработкой таких программ большие темпы развития набрало новое перспективное научное направление — компьютерная математика [20]. На рынок был выпущен целый ряд систем компьютерной математики (СКМ) — Derive [21], Mathcad [22, 23], Maple [24], Mathematica [25], MATLAB [26] и др. Здесь список литературных источников по СКМ ограничен лишь отдельными работами автора данной книги. Во всем мире по каждой СКМ опубликованы сотни книг, например только по MATLAB их более 600 (по данным сайта www.matlab.ru). Новые версии таких программ выходят практически ежегодно, а в их разработке принимают участие лучшие математические школы Запада и многие эмигрировавшие на Запад математики и программисты из бывшего СССР.

СКМ могут успешно применяться при решении многих задач математического моделирования, которые легко сводятся к достаточно простому математическому описанию. При этом они реализуют как численные, так и аналитические методы решения задач. Но автоматизация подготовки к решению сложных задач моделирования у СКМ практически отсутствует.

Число задач моделирования стало настолько большим, что специалистам потребовались достаточно универсальные системы блочного моделирования, реализующие визуально-ориентированный подход к имитационному моделированию произвольных по структуре, назначению и областям применения систем. Стало ясно, что такие системы нужно создавать как в виде отдельных

специализированных систем, так и на основе СКМ в виде их пакетов расширения.

Среди таких программ моделирования видное место заняла система блочного имитационного моделирования Simulink, которая интегрирована с мощной матричной системой компьютерной математики MATLAB [26—32]. Система MATLAB обеспечивает обширные возможности матричных вычислений, тогда как расширение Simulink позволяет на основе визуально-ориентированного программирования создавать крупные модели различных систем из устройств из отдельных блоков, автоматически составлять уравнения состояния моделей, решать их и наглядно представлять результаты моделирования.

Однако комплекс Simulink+ MATLAB+Toolbox+Blockset (Toolbox и Blockset — это наборы пакетов расширения системы Simulink+MATLAB [30—32]) оказался слишком громоздким для большинства применений. Так, последняя версия MATLAB 6.5 со всеми ее расширениями занимает на жестком диске ПК около 1,5 Гбайт. Кроме того, Simulink содержит чрезмерно большую библиотеку блоков, большинство их которых носит специализированный характер.

В связи с этим в последнее время резко возрос интерес к небольшой по объему (но достаточно мощной по возможностям) универсальной системе блочного имитационного визуально-ориентированного математического моделирования VisSim. Программа создана корпорацией Visual Solutuin Inc. (США). Основным разработчиком программы и главой корпорации является Петер Дарнелл (Peter Darnell). Наряду с самой системой VisSim выпущен ряд пакетов ее расширения, существенно повышающих и без того обширные возможности системы.

Эта жемчужина в мире программ математического моделирования давно привлекает интерес как специалистов в области математического моделирования, так и разработчиков систем компьютерной математики. Например, корпорация MathSoft, создатель знаменитой и самой массовой системы компьютерной математики Mathcad, не только обеспечила стыковку этой системы с программой VisSim, но и стала поставлять VisSim в составе некоторых версий СКМ Mathcad [23]. Версия VisSim 4.5 часто встречается на CD-ROM с системой Mathcad 2000/2001. Более того, VisSim может интегрироваться и с монстром среди СКМ — системой MATLAB+Simulink.

VisSim нашла широкое распространение в системе образования России и стран СНГ. Многие университеты давно применяют VisSim 3.0 для выполнения лабораторных работ по курсам автоматизированных систем управления, теории автоматического управления, электро- и радиотехники, математического моделирования в различных предметных областях и т. д. Большой вклад в распространение этой системы внес интернет-сайт vissin.nm.ru (создан под руководством Н. Климачева из Южно-Уральского университета). Вполне полноценная версия VisSim 3.0 распространяется бесплатно, ее можно найти и на CD-ROM, и даже на дискете, на которую она помещается в архивированном виде. А новейшая версия VisSim 5.0 со сроком бесплатной работы 60 дней может быть получена с интернет-сайта разработчика системы www.vissim.com (см. главу 2).

Данная книга впервые представляет описание трех последних версий системы VisSim 3/4.5/5 нашему читателю. В главе 1 книга содержит минимум теоретической информации о моделях и математическом моделировании. Глава 2 описывает интерфейс пользователя систем, а главы 3—5 являются справочным и практическим пособием по системе моделирования VisSim. При этом в главе 3 дано полное описание библиотеки блоков VisSim, в главе 4 описаны обработка и моделирование сигналов, в главе 5 — моделирование и проектирование систем. А глава 6 посвящена интеграции системы VisSim с известными офисными пакетами — СКМ Mathcad и MATLAB и пакетом расширения VisSim/Comm по проектированию и моделированию средств телекоммуникаций и связи.

В книге дается свыше 300 самых разнообразных примеров применения VisSim в моделировании и обработке сигналов, в моделировании и анализе самых различных систем и устройств. Это позволяет рекомендовать книгу как учебное пособие по математическому моделированию различных устройств, систем и явлений. Наряду с описанием включенных в поставку системы демонстрационных примеров дается ряд оригинальных примеров применения системы VisSim. При этом примерами применения охвачены все без исключения блоки библиотеки блоков системы VisSim, что делает книгу полноценным справочником по этой системе и руководством пользователя по ней.

Автор благодарит д. т. н., проф. В. В. Круглова и к. т. н., доц. А. А. Пенькова за сотрудничество по данной тематике и полезное обсуждение ряда относящихся к ней вопросов.

Книга ориентирована на инженеров и научных работников, занятых решением задач математического моделирования, а также на студентов и преподавателей вузов и университетов различного профиля, в которых изучаются основы теории и практики математического моделирования.

Глава 1
Введение в математическое моделирование

Математическое моделирование основано на достижениях математики — как классической, так и новейшей компьютерной математики, ориентированной на выполнение вычислений с помощью современных компьютеров [20]. Чтобы работать с такой мощной системой математического моделирования, как VisSim, нужен определенный минимум теоретических знаний по математики [35—36] и математическому моделированию [1—12]. Он и содержится в данной главе.

1.1. Основные понятия 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, достаточно отметить работы [1—12, 27—33].

С моделями и моделированием мы сталкиваемся в нашей жизни каждый день. В детстве ребенка окружают игрушки — машинки, куклы, конструкторы и т. д. — модели, повторяющие отдельные свойства реально существующих предметов. Играя, ребенок получает важные знания о них и, вырастая, начинает грамотно применять уже реальные объекты. В процессе мышления человек оперирует образами объектов окружающего мира, которые являются разновидностями моделей — когнитивными (мысленными) моделями. В сущности произведения живописи, скульптуры и литературы тоже можно считать моделями реальных объектов.

Исключительно велика роль моделирования в ядерной физике и энергетике. Достаточно отметить, 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, то означало бы уничтожение жизни на Земле. Запрет на испытания ядерного оружия стал возможен также благодаря самым изысканным средствам моделирования ядерных и термоядерных процессов. Трудно пере-

оценить 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 изделий. Чем сложнее проектируемый объект, тем, как правило, важнее роль моделирования в его изучении и создании. Самое широкое применение моделирование находит в 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 лабораторные работы заменяются их компьютерным моделированием. Автор данной книги не относится к почитателям такого подхода, но нельзя не отметить, что современные программы моделирования настолько хорошо моделируют реальные процессы, устройства и системы, что появляется иллюзия и впрямь работы с реальными устройствами. И это широко используется на практике — достаточно вспомнить тренажеры для космонавтов, летчиков и танкистов, широко используемые при обучении военному мастерству. Но, пожалуй, главное заключается в том, что математическое моделирование позволяет понять физическую и математическую сущность моделируемых явлений и обосновать оптимальные подходы к проектированию самых различных изделий.

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

1) модель должна быть адекватной оригиналу в том смысле, что должна с достаточной точностью отображать интересующие исследователя характеристики оригинала;

2) модель должна устранять проблемы, связанные с физическим измерением каких-то сигналов или характеристик оригинала.

Следует отметить, что в большинстве случаев моделирование вовсе не заменяет реальный объект и не отменяет необходимости в его разработке и натурном испытании. Оно просто значительно уменьшает объем работ по проектированию и исследованию объектов. В тех же случаях, когда это не так, стоимость моделирования может оказаться вполне сравнимой со стоимостью разработок и натурных испытаний изделий.

1.2. Основные виды моделей и их свойства

1.2.1. 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.

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.

1.2.2. 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*) — тем 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.

1.3. Цели, принципы и технология моделирования

1.3.1. Цели моделирования

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

1) изучение механизма явлений (познавательная цель);

2) 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.

1.3.2. Понятие о сигналах

Во многих случаях целью моделирования является изучение реакции системы или устройства на некоторые воздействия, в качестве которых обычно 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 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 (например, электрические сигналы можно преобразовать в оптические, и наоборот) при сохранении имеющейся в сигнале информации.

С помощью источников сигналов можно оценивать поведение различных устройств и систем. К примеру, важнейшие характеристики линейных усили-

телей 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 $+\infty$, т. е., по существу, он является теоретической абстракцией (достаточно отметить, что энергия такого сигнала равна бесконечности). Естественно, что на практике сигнал такого вида рассматривается в конечном интервале времени.

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.

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 про-

сто для получения информации о закономерностях работы тех или иных систем и устройств.

1.3.3. Основные принципы моделирования

Моделирование базируется на нескольких основополагающих принципах. Рассмотрим их.

Принцип информационной достаточности

При полном отсутствии информации об исследуемом объекте построение его модели невозможно. С другой стороны, при наличии полной информации об объекте построение его модели не имеет смысла. Существует некоторый уровень априорной информации об объекте, при достижении которой может быть построена его адекватная модель.

Принцип осуществимости

Создаваемая модель должна обеспечивать достижение поставленной цели исследования с вероятностью, существенно отличающейся от нуля.

Принцип множественности моделей

Данный принцип является ключевым. Речь идет о том, что создаваемая модель должна отражать в первую очередь те свойства реальной системы, которые интересуют исследователя. Соответственно, при использовании любой конкретной модели познаются лишь некоторые стороны реальности. Для более полного ее исследования необходим ряд моделей, позволяющий с разных сторон и с разной степенью детализации рассмотреть исследуемый объект.

Принцип агрегирования

В большинстве случаев сложную систему можно представить состоящей из агрегатов (подсистем), для адекватного математического описания которых оказываются пригодными некоторые стандартные математические схемы.

Принцип параметризации

Этот принцип означает, что модель строится в виде известной системы, параметры которой неизвестны.

1.3.4. Технология моделирования

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. Соответ-

ствующие 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.

1.3.5. 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:

1) *графические* — оценочные приближенные методы, основанные на построении и анализе графиков;

2) *аналитические* — решения, полученные строго в виде аналитических выражений (пригодны для узкого круга задач);

3) *численные* — основной инструмент для решения сложных математических задач, основанный на применении различных численных методов.

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 — неустранимые 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.

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, т. е. 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:

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

где ν_Δ — абсолютное число обусловленности ν_δ — относительное число обусловленности. Для плохо обусловленных задач $\nu_\delta \gg 1$, 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 $\delta(x^*) \le 0.0001$ %, то при $\nu_\delta = 10^3$ — задача хорошо обусловлена ($\delta(y^*) = 0.1$ %).

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 $f'(x) \approx \frac{f(x+h) - f(x)}{h}$.
- *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 части 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:

1) ее решения существует при любых допустимых входных данных;
2) это решение единственно (однозначно определено);
3) 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; т. е. 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. Пример задачи моделирования полета камня

1.4.1. Постановка задачи моделирования

Началу моделирования предшествует постановка содержательной задачи моделирования, переход от когнитивной модели к формулировке в словесной форме основных вопросов об объекте моделирования. Правильная постановка задачи очень важна, так как ошибка здесь потребует вернуться к построению модели с самого начала. Содержательная постановка задачи, называемая в технических дисциплинах *техническим заданием*, в дальнейшем уточняется и конкретизируется, однако принципиальные, основные положения остаются неизменными.

В качестве примера рассмотрим постановку типичной задачи моделирования «Бросок камня», позволяющую описать полет камня, брошенного под углом к горизонту. Эта задача давно используется как показательная при моделировании физических явлений [22, 33]. Сформулируем требования к модели и исходные данные для моделирования.

Модель должна позволять:
вычислять положение камня в любой момент времени.

Исходные данные:
масса камня, начальные координаты, начальная скорость и угол броска мяча.

1.4.2. Концептуальная формулировка задачи

На основе содержательной модели разрабатывается концептуальная формулировка задачи моделирования. Применительно к нашей задаче движение камня может быть описано в соответствии с законами классической механики Ньютона.

Гипотезы, принятые для модели:

- камень будем считать материальной точкой массой m, положение которой совпадает с центром масс камня;
- движение происходит в поле силы тяжести с постоянным ускорением свободного падения g и описывается уравнениями классической механики Ньютона;
- движение камня происходит в одной плоскости, перпендикулярной поверхности Земли;
- сопротивлением воздуха на первых порах пренебрегаем.

В качестве параметров движения будем использовать координаты (x, y) и скорость $v(v_x, v_y)$ центра масс камня.

Концептуальная постановка задачи на основе принятых гипотез заключается в определении закона движения материальной точки массой m под действием силы тяжести, если известны начальные координаты точки x_0 и y_0, ее начальная скорость v_0 и угол броска α_0.

Таким образом, модель является простой — объект, как материальная точка, не имеет внутренней структуры. Учитывая типичные скорости и высоту броска камня, можно считать постоянным ускорение свободного падения. Переход от трехмерных координат к плоскости значительно упрощает решение задачи. Он вполне допустим, если камень не подкручивается при броске. Пренебрежение сопротивлением воздуха, как будет показано далее, приводит к значительной систематической ошибке результатов моделирования.

1.4.3. Построение математической модели

Теперь перейдем к составлению *математической модели* объекта — совокупности математических соотношений, описывающих его поведение и свойства. Из законов и определяющих выражений предметной дисциплины формируются уравнения модели.

По оси x на камень не действуют никакие силы, по оси y действует сила тяжести. Согласно законам Ньютона имеем уравнения движения по оси x и оси y:

$$m \cdot \frac{d^2x}{dt^2} = 0,\ m \cdot \frac{d^2y}{dt^2} = -m \cdot g,\ v_x = \frac{dx}{dt},\ v_y = \frac{dy}{dt} \qquad (1.1)$$

при следующих начальных условиях

$$x(0) = x_0,\ y(0) = y_0,\ v_x(0) = v_0 \cdot \cos\alpha_0,\ v_y(0) = v_0 \cdot \sin\alpha_0.$$

Надо найти зависимости $x(t)$, $y(t)$, $v_x(t)$, $v_y(t)$.

Математическая постановка решения задачи в нашем случае соответствует решению задачи Коши для системы обыкновенных дифференциальных

уравнений с заданными начальными условиями. Известно, что решение задачи Коши существует и что оно единственное. Количество искомых переменных равно количеству дифференциальных уравнений. Таким образом, математическая модель корректна. Решение этой задачи есть в любом учебнике физики.

1.4.4. Выбор метода решения

Наша задача может быть решена как аналитически, так и численно. Рассмотрим оба варианта.

Аналитическое решение

Из (1.1) запишем систему ОДУ первого порядка:

$$\frac{dv_x}{dt} = 0,\ \frac{dv_y}{dt} = -g,\ v_x = \frac{dx}{dt},\ v_y = \frac{dy}{dt}. \qquad (1.2)$$

После интегрирования получим:

$$v_x(t) = C_1,\ v_y(t) = C_2 - g \cdot t,\ x(t) = C_3 + C_1 \cdot t,\ y(t) = C_4 + C_2 \cdot t - \frac{g \cdot t^2}{2}. \qquad (1.3)$$

Определив константы интегрирования из начальных условий, окончательно запишем:

$$x(t) = x_0 + v_0 \cdot \cos\alpha_0 \cdot t, \quad y(t) = y_0 + v_0 \cdot \sin\alpha_0 \cdot t - \frac{g \cdot t^2}{2},$$

$$v_x(t) = v_0 \cdot \cos\alpha_0, \qquad v_y(t) = v_0 \cdot \sin\alpha_0 - g \cdot t.$$

Из аналитического решения вытекает, что полет камня при отсутствии сопротивления воздуха происходит строго по параболической траектории, причем она на участках полета камня вверх и вниз симметрична. Необходимые для расчета уравнения заданы в параметрической форме — как зависимости от времени, что, кстати говоря, облегчает моделирование по ним полета камня.

Численное решение конечно-разностным методом

Численное решение может быть найдено только для конкретных значений параметров модели, например $m = 200$ г, $\alpha_0 = 45°$ $v_0 = 20$ м/с, $g = 9.8$ м/с^2, $x_0 = 0$, $y_0 = 1$ м. Существует большое количество численных методов решения систем ОДУ. Для данной задачи можно использовать простейший явный метод Эйлера, который является разновидностью конечно-разностных методов

Пусть дифференциальное уравнение приведено к виду $\frac{dy}{dx} = f(x, y)$, а вид функции $f(x, y)$ известен. Заменим приближенно дифференциалы приращениями, тогда

$$\Delta y = f(x, y) \cdot \Delta x \text{ и } y^{n+1} = y^n + f(x^n, y^n) \cdot \Delta x.$$

Аналогично, для системы ОДУ данной задачи получим расчетные формулы:

$$v_x^{n+1} = v_x^n, \qquad v_y^{n+1} = v_y^n - g \cdot \Delta t,$$
$$x^{n+1} = x^n + v_x^n \cdot \Delta t, \qquad y^{n+1} = y^n + v_y^n \cdot \Delta t.$$

Очевидно, что вычисления надо вести до момента времени t_k, когда $y(t_k)$ станет равным 0, камень упадет на землю. Подобное решение ввиду его приближенного характера целесообразно только в том случае, если используемая система моделирования не имеет средств достаточно точного решения систем дифференциальных уравнений или когда просто нужна демонстрация простых методов решения задачи.

В более сложных случаях выбор численного метода решения является ответственным этапом, необходимо учитывать жесткость системы ОДУ, скорость работы, сходимость и точность метода.

1.4.5. Программная реализация модели на ЭВМ

Существует довольно много программных средств, позволяющих реализовать задачи математического, в частности имитационного, моделирования. Это и языки программирования различного уровня, и специализированные и универсальные программные средства.

Языки программирования и специализированные программные средства используются обычно при решении небольшого числа серьезных задач моделирования. Имея высокую скорость моделирования, они обеспечивают малое время собственно моделирования. Однако подготовка к нему оказывается крайне трудоемкой и дорогой. Поэтому часто предпочитают применять универсальные программные средства с малым временем подготовки задач.

Отметим следующие группы универсальных программных средств.

- Пакеты блочного моделирования, например Simulink в системе MATLAB или описанная в этой книге программа визуально-ориентированного моделирования VisSim.
- Пакеты физического моделирования, например 20SIM Pro.
- Универсальные системы компьютерной математики Mathcad, Maple, Mathematica, MATLAB и др.
- Прочие программные системы, например статистические пакеты GPSS, Statistica и др.

Среди систем компьютерной математики наиболее простой в освоении и имеющей наиболее удобный пользовательский интерфейс представляется система Mathcad [21—23]. Хотя прямо она не ориентирована на задачи имитационного моделирования, они успешно решаются в ее среде. В качестве примера возьмем нашу задачу.

Воспользуемся СКМ Mathcad 2001i. Особенность ее работы — максимальное приближение записи алгоритма решения к естественной математической форме. Разработчики системы стремились обеспечить решение большинства задач без использования программирования. Хотя с помощью этой математической системы возможно аналитическое решение некоторых задач моделирования, мы ограничимся численным решением.

Система Mathcad имеет несколько встроенных функций для решения систем ОДУ (rkadapt, Odesolve и др). При их использовании достаточно корректно записать условие задачи и вызвать соответствующую команду (см. рис. 1.1), на котором показано решение задачи на полет брошенного камня в условиях отсутствия сопротивления воздуха с помощью функции Odesolve).

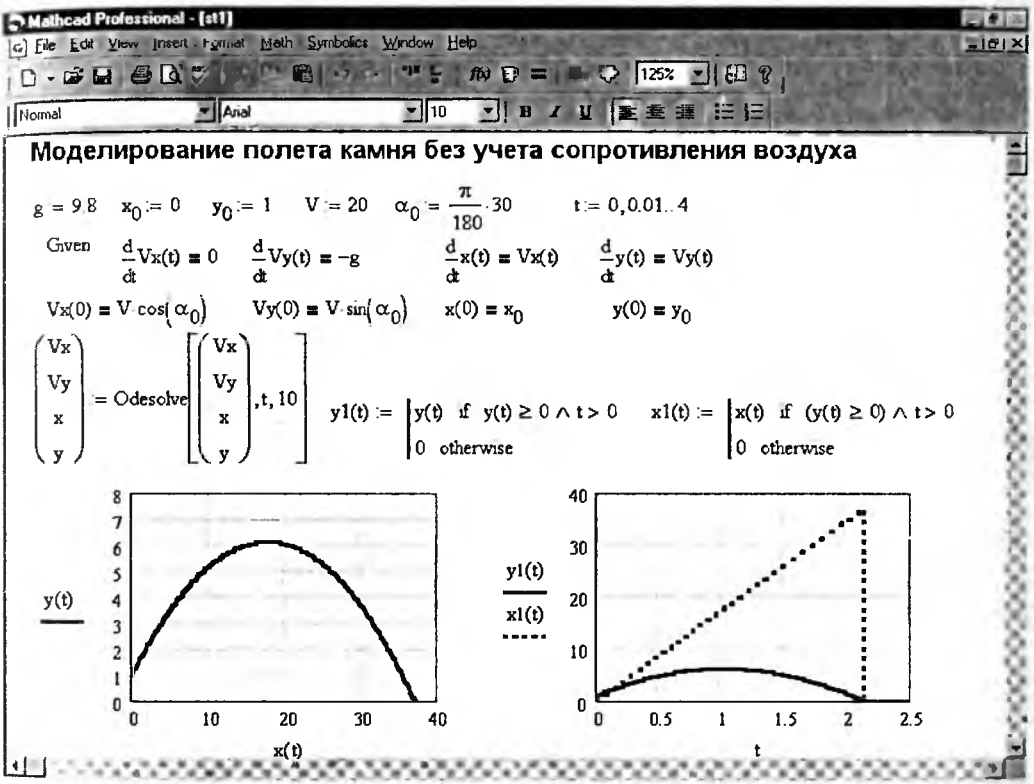


Рис. 1.1. Программная реализация численного решения в среде Mathcad задачи на полет камня без учета сопротивления воздуха

Как и следовало ожидать, траектория полета камня оказывается квадратичной параболой. Если пользователь желает глубже ознакомиться с реализацией численных методов моделирования, он может составить свою реализацию подходящего метода, например Эйлера или Рунге—Кутта. Информацию о них можно найти в любом учебнике по численным методам.

1.4.6. Проверка адекватности модели

Необходимым требованием, которому должна отвечать каждая модель, является адекватность — соответствие результатов, полученных при моделировании, данным эксперимента, теоретическим положениям или тестовым примерам. Ограничимся нашим примером.

Результаты, полученные по приведенной выше модели, будут существенно отличаться от действительных. Особенно это будет заметно, если умень-

шать вес камня, например, взяв вместо камня аналогичных размеров кусок пенопласта. Очевидно, это можно объяснить только грубым упрощением в принятой системе гипотез — пренебрежением сопротивления воздуха. Учет этого фактора требует изменения концептуальной модели и всех последующих этапов решения, а именно требуется учитывать сопротивление воздуха.

Давайте уточним математическую модель. Сила сопротивления воздуха направлена против направления движения камня:

$$m \cdot \frac{d^2x}{dt^2} = -Fmp_x, \quad m \cdot \frac{d^2y}{dt^2} = -m \cdot g - Fmp_y, \quad v_x = \frac{dx}{dt}, \quad v_y = \frac{dy}{dt}.$$

Сопротивление воздуха зависит от скорости движения тела и может быть описано следующей эмпирической формулой:

$$Fmp = A \cdot v + B \cdot (v)^3,$$

где $A = 0.1$ Н · с/м, $B = 10^{-3}$ Н · с^3/м^3. Введение этого соотношения делает дифференциальные уравнения и основанную на них модель нелинейными.

Поскольку нелинейные задачи в аналитическом виде чаще всего не решаются, выберем численный метод, который легко реализуется функциями Mathcad. На рис. 1.2 представлено решение данной задачи в среде системы Mathcad.

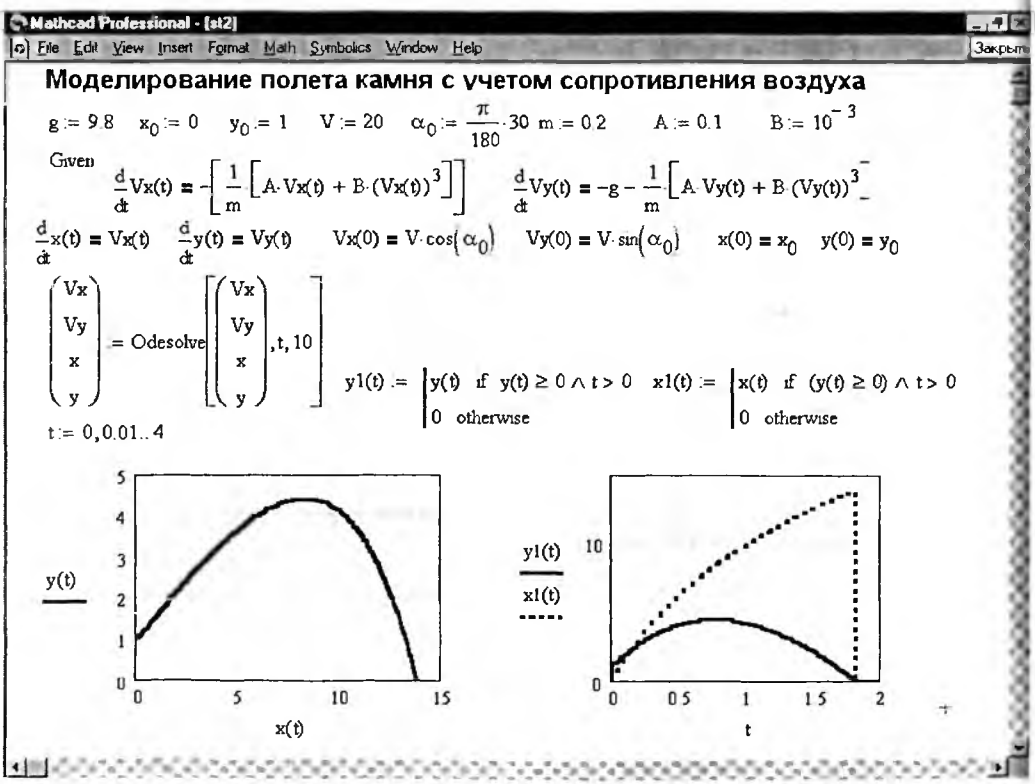


Рис. 1.2. Моделирование полета камня с учетом сопротивления воздуха

В целом надо сказать, что решение такой простой задачи в среде математической системы Mathcad вполне оправдано и достаточно наглядно. Правда, не совсем ясна взаимосвязь между математическими объектами в документе Mathcad, но она соответствует простому правилу — блоки исполняются слева направо и сверху вниз. Разумеется, подобная невидимая связь между блоками совершенно неприемлема для представлений о структуре мало-мальски сложных моделей, где связь между блоками бывает куда более сложной и подчас запутанной. Кроме того, современные принципы модульного программирования предполагают, что в состав моделей (блоков) могут входить подмодели (субблоки), что сильно усложняет структуру моделей, да и строго определенное расположение блоков чаще всего просто не приемлемо.

1.4.7. Анализ результатов моделирования

Анализ результатов моделирования — необходимый этап грамотного решения любой задачи. Такой анализ позволяет:

- получить представление о поведении объекта в различных условиях, найти оптимальные характеристики процесса;
- определить область применения модели;
- оценить обоснованность принятых при построении модели гипотез, определить пути ее совершенствования.

Графический анализ результатов решения задачи численным методом показан на рис. 1.2. Из приведенного примера явно видно, что при учете сопротивления воздуха траектория полета камня заметно отличается от параболической, она заметно круче на спаде, и камень пролетает меньшее расстояние.

Итак, мы рассмотрели классическую задачу на полет камня, которая свелась к решению системы дифференциальных уравнений, описывающих такой полет. Поначалу это была система линейных дифференциальных уравнений, но при учете сопротивления воздуха система стала нелинейной.

1.5. Некоторые замечания о моделировании

1.5.1. Недостатки моделирования с помощью систем компьютерной математики

Главным достоинством моделирования с помощью систем компьютерной математики (Mathcad, например) является математическая прозрачность вычислений и легкость создания объектов, осуществляющих математические вычисления — даже самые сложные. Решение задачи при этом заключается в вводе в систему необходимых формул и учете не слишком сложных особенностей синтаксиса описания алгоритмов нужных вычислений.

Однако такому подходу к моделированию присущ целый букет недостатков. Основные из них следующие:

- математическое описание модели приходится выполнять «вручную» самим разработчиком модели, что возможно только для очень простых моделей;

- плохо просматриваются или вообще не просматриваются связи между отдельными этапами моделирования;
- строго заданное расположение блоков, нужное для их правильного исполнения, в сложных моделях неприемлемо;
- отсутствует возможность блочного моделирования с применением достаточно большого числа типовых блоков для реализации тех или иных задач моделирования;
- как только возникает необходимость ввести в модель описание нового блока, его приходится вводить заново;
- отсутствует функциональная схема (блок-схема или диаграмма) модели, которую нередко и считают собственно моделью;
- плохо выделены элементы ввода и вывода;
- нет виртуальных реализаций многих измерительных приборов, к которым привыкли ученые экспериментаторы и инженеры (и даже радиолюбители);
- отсутствует возможность быстрой модификации модели, например повышение порядка модели может потребовать ее задания заново;
- отсутствует возможность оперативного изменения метода моделирования в целом или осуществления базовой операции интегрирования;
- часто явно недостаточна скорость моделирования.

Это достаточно серьезные недостатки, ограничивающие применение систем компьютерной математики в моделировании решением простых задач и задач, в которых главным аспектом является математическая прозрачность решения. В других случаях возникает необходимость в применении специально ориентированных на моделирование программных средств. Подчас при этом математическая прозрачность решения начисто утрачивается и пользователь может вести моделирования вообще не представляя, каким образом его результаты получаются; примером могут служить программы схемотехнического моделирования.

Система VisSim здесь занимает золотую серединку. Эта система имеет типовые библиотечные блоки, из которых может легко собираться модель как простая, так и самая сложная, возможно содержащая множество подблоков (субблоков). В то же время назначение подавляющего большинства блоков математически вполне прозрачно. При небольшом навыке вполне очевидны и правила составления графических моделей.

1.5.2. О моделировании задач управления

К широко распространенным задачам математического моделирования относятся задачи из области автоматического управления [31, 40]. По существу, практически все задачи моделирования можно свести к задачам управления. Например, рассмотренную выше задачу на полет камня можно считать задачей управления полетом камня, особенно, если усложнить ее и принять, что полет камня должен быть таким, чтобы камень попал в заданное место или в заданную цель или поразил ее с заданной степенью вероятности. В теории управления широкое распространение получили математические модели

объектов управления, используемые как для анализа, так и для синтеза систем регулирования.

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

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

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

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

1.5.3. Понятие о динамических объектах

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. 1.3, 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).

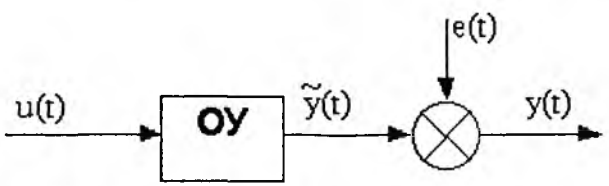


Рис. 1.3. Общее представление идентифицируемого объекта О

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.

Иногда задачу определения математической зависимости можно решить чисто аналитическим путем. Примером может служить простая RC-цепь, показанная на рис. 1.4.

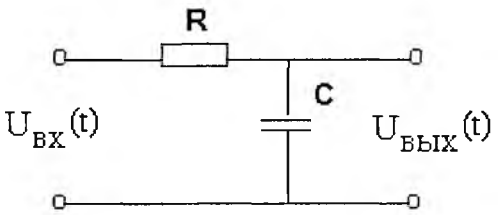


Рис. 1.4. Простая RC-цепь

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

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

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

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. Отметим, что при экспериментальном определении параметров модели необходимо обеспечить:

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

1.5.4. 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).

С учетом выполнения принципа суперпозиции можно выделить два случая, относящихся к линейности объектов.

1. Объект линеен по входному воздействию:

$$\tilde{y}(t) = \Psi[\beta, u_1(t) + u_2(t)] = \Psi[\beta, u_1(t)] + \Psi[\beta, u_2(t)] = \tilde{y}_1(t) + \tilde{y}_2(t).$$

2. Объект линеен по параметрам:

$$\tilde{y}(t) = \Psi[\beta_1 + \beta_2, u(t)] = \Psi[\beta_1, u(t)] + \Psi[\beta_2, u(t)] = \tilde{y}_1(t) + \tilde{y}_2(t).$$

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

1.5.5. Понятие об идентификации систем

Для лучшего понимания целей и задач математического моделирования полезно уточнить понятие *идентификации систем* [31, 40]. Под идентификацией динамических объектов понимают процедуру определения структуры и параметров их математических моделей, которые при одинаковых входном сигнале объекта и модели обеспечивают близость выхода модели к выходу объекта при наличии какого-то критерия качества.

Обычно идентификация — многоэтапная процедура. Основные ее этапы следующие.

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

Отметим, что в связи с многообразием объектов и различных подходов к их моделированию существует множество вариантов задачи параметрической идентификации. Дальнейшее изложение будет относиться к линейным стационарным динамическим объектам.

1.6. Виды моделей объектов управления и их характеристики

Рассмотрим основные виды моделей линейных непрерывных стационарных динамических объектов и их взаимосвязь (действием шума $e(t)$ пока пренебрегаем) [30, 31].

1.6.1. Дифференциальное уравнение

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 системы, нередко имеют аналитические решения. Иногда их можно найти в аналитическом виде, например используя функции символьных вычислений системы Mathcad и других систем 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 системе VisSim.

1.6.2. Передаточная функция

При моделировании линейных систем часто используется операторный метод. Особенно широко он применяется в электро- и радиотехнике. Например, в линейных цепях замена применение 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 $L\{\bullet\}$ — 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 фирменном описа-

нии VisSim используется оператор Лапласа s, коэффициенты полинома числителя обозначены буквой a, а знаменателя — буквой b. Начиная с главы 3 такая неоднозначность будет решаться в пользу принятых в VisSim правил.

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. VisSim, в частности, имеет очень простые и эффективные средства для этого. Изучая положение нулей и полюсов на комплексной плоскости, можно судить о свойствах системы.

1.6.3. Импульсная характеристика $w(t)$

Под *импульсной характеристикой* (ИХ) понимается реакция предварительно невозмущенного объекта (т. е. 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 нереализуема, тем не менее импульсная характеристика линейных систем и устройств является вполне реальной характеристикой, широко используемой на практике. Многие системы моделирования, в том числе VisSim, имеют средства для ее построения.

1.6.4. 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:

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

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 и VisSim.

1.6.5. 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.

1.6.6. 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:

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

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:

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

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:

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

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, т. е. вычисление коэффициентов Фурье, принято назвать *спектральным анализом*, а воссоздание приближения функции рядом Фурье, т. е. 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 $f_k = kf_1$, 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). Не повторяя общеизвестного описания реализаций БПФ [39], отметим, что функции, реализующие прямое и обратное БПФ, есть как в системе VisSim, так и в системе Mathcad. Они предоставляет возможность проводить указанные преобразования для данных в виде векторов как с действительными, так и комплексными элементами.

У системы Mathcad функция *fft(v)* выполняет БПФ для данных, представленных действительными числами — значениями исходного вектора v. Он должен иметь $2m$ составляющих, где m — целое число. Элементы вектора, возвращаемого функцией *fft(v)*, соответствуют формуле:

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

Здесь n — число элементов вектора v, 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 функцией *fft(v)*, — это в общем случае комплексные числа, даже если сигнал представлен вещественными отсчетами.

Функция *ifft*(*v*) реализует обратное (инверсное) преобразование Фурье для вектора *v* с комплексными элементами. Вектор *v* здесь должен иметь $1 + 2m + 1$ элементов. Функция *ifft*(*v*) вначале создает вектор *w*, комплексно-сопряженный с *v*, и затем присоединяет его к вектору v. После этого вычисляется вектор d с элементами, рассчитанными по следующей формуле:

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

Функции *fft*(*v*) и *ifft*(*v*) дают точные (в пределах погрешности численных расчетов) обращения. При этом *ifft*(*fft*(*v*)) = *v*, что можно использовать для проверки преобразований.

Функция *cfft*(*A*) аналогична предыдущей, но реализует прямое преобразование Фурье для вектора *A* с комплексными элементами. Если *A* — матрица, реализуется двумерное преобразование. Введение функции *fft*(*v*) обусловлено тем, что преобразование для векторов с действительными элементами реализуется по более быстрому алгоритму (БПФ) и занимает меньше времени. Кроме того, в этом случае проще ввод исходных данных.

Функция *icfft*(*v*) выполняет обратное преобразование Фурье по полному алгоритму, при котором как исходный, так и результирующий векторы или матрицы содержат элементы с комплексными значениями.

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

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

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).

Альтернативные формулы преобразований Фурье в системе Mathcad используются в функциях FFT(v), IFFT(v), CFFT(v) и ICFFT(v). В остальном использование этих функций не отличается от аналогичных функций *fft*(*v*), *ifft*(*v*), *cfft*(*v*) и *icfft*(*v*). Надо лишь помнить о том, что нормировка функций БПФ в литературе может быть различной. Это нередко создает путаницу при проведении операций спектрального анализа и синтеза и приводит к разным результатам.

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

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

Соответственно, обратное преобразование Фурье задается следующим образом:

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

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.

В системе VisSim имеются блоки fft и ifft, реализующие аналогичные функции прямого и действительного преобразований Фурье для векторов действительных данных. Возможно также оперативное построение спектров любых функций, представленных в окнах виртуального графопостроителя (осциллографа). Расширенная реализация этих преобразований для векторов с комплексными данными имеется в пакета расширения VisSim/Com. В целом средства Фурье-анализа и синтеза в VisSim доведены до высокой степени совершенства и позволяют выполнять Фурье-преобразования любого вида с прекрасной графической их визуализацией.

1.6.7. Частотные характеристики

Частотные характеристики объекта определяются его комплексным коэффициентом передачи $W(j\omega) = W(p) \mid p = j\omega$, который является Фурье-преобразованием ИХ. В радиотехнике и электронике вместо коэффициента передачи пользуются понятием комплексного коэффициента усиления $K(j\omega) = K(\omega)$ [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 ∞, т. е. 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.

1.6.8. 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 является выделение и описание *состояний* системы. Каждое состояние характеризуется набором значений некоторых переменных, называемых переменными состояния [12].

При выборе n координат системы (объекта) в качестве переменных ее состояния (такими координатами, например, могут быть выходной сигнал $y(t)$ и

$n-1$ его производных $x_i(t)$, $i = 1, 2, ..., n$ данную систему можно описать *уравнениями для переменных состояния*:

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

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

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

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

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

где **I** — единичная матрица.

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

1.6.9. 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 и 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

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

через дискретную передаточную функцию

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

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

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

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

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), т. е.

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

Для дискретных объектов также может быть использовано описание через переменные состояния

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

$$y_k = \mathbf{C}\mathbf{X}_k + \mathbf{D}u_k,$$

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

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

1.7. Понятия статистического моделирования

1.7.1. 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 [35] и 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 $\varnothing$ 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 лежит 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 равновозможные варианты 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 $P^*(A) = m/n$ называют 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 систему VisSim. 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 данные используются в порядке их возрастания.

1.7.2. Решение задач комбинаторики

К числу элементарных задач статистики относятся задачи комбинаторики. Рассмотрим основные из них.

Перестановкой n объектов называют их расположение в определенном порядке. Число перестановок задается как значение факториала $P_n = n \cdot (n-1) \times \times \ldots \cdot 2 \cdot 1 = n!$, для вычисления которого в Mathcad есть оператор !. Например,

число перестановок для 10 предметов есть 10! = 3628800. Значения факториала быстро растут с ростом n.

Размещением некоторой части m из множества n элементов называется их расположение в определенном порядке. Число размещений обозначают как

$$A_n^m = n \cdot (n-1) \cdot \ldots \cdot (n-m) \cdot (n-m+1) = \frac{n!}{(n-m)!}.$$

Пример: сколько вариантов набора двух разных цифр возможно на диске телефона, имеющим 10 цифр? Ответ: $A_{10}^2 = 10 \cdot 9 = 90$.

Сочетанием m элементов из множества n элементов называют любую часть элементов (подмножества) этого множества

$$C_n^m = \frac{A_n^m}{m!}.$$

Пример: сколько способов выбора делегации из $m = 3$ человек возможны из группы, насчитывающей $n = 10$ человек. Имеем

$$C_{10}^3 = \frac{10 \cdot 9 \cdot 8}{1 \cdot 2 \cdot 3} = 120.$$

СКМ, как правило, имеют расширенный набор функций комбинаторики. А в системах математического моделирования эти функции могут быть легко вычислены по приводимым для них формулам.

1.7.3. Дискретные и непрерывные случайные величины

В теории вероятностей *случайной величиной* называют переменную величину, которая в зависимости от исхода испытания случайно принимает какое-либо одно значение из множества возможных значений. Случайные величины могут быть дискретными и непрерывными. Примерами дискретных случайных величин является последняя цифра номера телефона или число студентов в группе.

Дискретные случайные величины задаются своими значениями и их вероятностями, например в виде следующей таблицы:

X	x_1	x_2	x_3	...	x_{n-1}	x_n
P	p_1	p_2	p_3	...	p_{n-1}	p_n

В сумме вероятности дискретной случайной величины равны 1. Математическим ожиданием дискретной случайной величины называют значение $M(X) = x_1 \cdot p_1 + x_2 \cdot p_2 + \ldots + x_n \cdot p_n$. Математическое ожидание ряда дискретных случайных величин приближенно равно среднему значению, т. е. $M(X) \approx \overline{x} = (x_1 + x_2 + \ldots x_n)/n$.

Мерой «рассеивания» дискретных случайных величин могло бы служить *отклонение* случайных величин от их математического ожидания. Но, имея разные знаки, отклонения часто взаимно компенсируются. Поэтому мерой «рассеивания» принято считать *квадрат отклонений* случайной величины X

(вы, вероятно, подметили, что большими буквами обозначаются случайные величинами, а малыми из значения).

Дисперсией $D(X)$ дискретной случайной величины X называется математическое ожидание квадрата отклонения случайной величины X от ее математического ожидания, т. е. $D(X) = M[X - M(X)]^2$. А *средним квадратичным отклонением* называют корень квадратный из дисперсии:

$$\sigma(X) = \sqrt{D(X)}.$$

Непрерывные случайные величины могут принимать любые значения на том или ином отрезке. Поэтому их закон распределения нельзя описать в виде таблицы. *Функцией распределения* (или интегральной функцией распределения) непрерывных случайных величин называется функция $F(x)$, равная вероятности того, что случайная величина X приняла значение, меньшее x: $P(X) = \{(X < x)$. Функция $F(x)$ всегда монотонно растущая и ее значения лежат в пределах от 0 до 1. Если значения x лежат в пределах от $-\infty$ до $+\infty$, то $F(-\infty) = 0$ и $F(+\infty) = 1$.

Плотностью вероятности (или дифференциальной функцией распределения) случайной величины называют функцию $f(x) = F'(x)$. Вероятность попадания непрерывной случайной величины в интервал $(a; b)$ равна интегралу от $f(x)$ в пределах от a до b:

$$P(a < x < b) = \int_a^b f(x)dx = F(b) - F(a).$$

Математическим ожиданием $M(X)$ непрерывной случайной величины X с плотностью вероятности $f(x)$ называют величину несобственного интеграла:

$$M(X) = \int_{-\infty}^{\infty} x \cdot f(x)dx,$$

а дисперсией случайной величины с математическим ожиданием a именуют величину:

$$D(X) = \int_{-\infty}^{\infty} (x - a)^2 f(x)dx.$$

1.7.4. Законы распределения и статистические функции

В статистической обработке данных генеральной или выборочной совокупностей используются различные законы распределения непрерывных случайных величин. Один из самых простых законов — равномерный. Он соответствует постоянному значению $f(x) = C$ на отрезке $[a, b]$ с единичной площадью зависимости $f(x)$. Отсюда следует, что $C = 1/(b - a)$. Система Mathcad, к примеру, имеет функцию rnd(x) для генерации случайных чисел с равномерным распределением на отрезке $[0, x]$.

Одним из самых распространенных является нормальный закон распределения:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-a)^2}{2\sigma^2}},$$

где a — математическое ожидание; σ — среднее квадратичное отклонение. Интегральная функция распределения для него:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(x-a)^2}{2\sigma^2}}.$$

Система VisSim имеет весьма ограниченный набор средств для проведения статистического моделирования. По аналогии с задачей на моделирование карточной игры реализованные методы называют методами Монте-Карло. Среди источников сигналов в VisSim есть всего два генератора случайных чисел — с равномерным и нормальным законами распределения.

Остальные законы можно реализовать либо известными преобразованиями этих законов, либо применением куда более обширного набора средств для статистического моделирования, которые встроены в систему Mathcad, легко интегрируемую с VisSim. Кроме того, VisSim имеет генератор случайной последовательности дискретных данных. Все эти средства будут рассмотрены в главе 3.

В то же время в интегрируемой с VisSim системе Mathcad имеется ряд наборов функций, относящихся к наиболее распространенным законам распределения. Характер функции для каждого закона распределения (их 18) задается первой буквой их имени:

d (Density) — плотность вероятности $f(x)$;

p (Probality) — функция распределения $F(x)$;

q (Quantil) — инверсная функция распределения — *квантиль*;

r (Random) — вектор случайных чисел.

Квантили функций распределения случайных величин позволяют по заданной вероятности вычислить такое значение x, при котором вероятность равна или меньше заданного значения p. А функции, начинающиеся с буквы r, служат для генерации случайных чисел с заданным законом распределения.

В качестве примера приведем функции нормального и экспоненциального распределений:

dnorm(x,μ,σ) pnorm(x,μ,σ) qnorm(p,μ,σ) rnorm(m,μ,σ) — нормальное распределение ($\mu = a$ — среднее значение, $\sigma > 0$ — среднеквадратичное отклонение);

dexp(x,r) pexp(x,r) qexp(p,r) rexp(m,r) — экспоненциальное распределение (r, $x > 0$).

Кроме того, Mathcad легко обеспечивает выполнение многих *статистических расчетов*. К широко распространенным статистическим функциям общего характера в системе Mathcad относятся следующие функции скалярного аргумента x:

cnorm(x) — кумулятивная нормальная функция — подобна функции pnorm (x,0,1), описанной выше;

erf(x) — функция ошибок (или интеграл вероятности)

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt;$$

cerf(x) — дополнительная функция ошибок $(1 - erf(x))$.

Следующая группа функций относится к вычислению основных статистических параметров одного массива данных (матрицы размера $m \times n$ или вектора):

mean(A) — возвращает среднее значение элементов массива **A**:

$$mean = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A_{i,j};$$

gmean(A) — возвращает гармоническое среднее значение элементов массива **A**:

$$gmean = (\prod_{i=0}^{m-1} \prod_{j=0}^{n-1} A_{i,j})^{1/(mn)};$$

hmean(A) — возвращает геометрическое среднее значение элементов массива **A**:

$$mean = (\frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \frac{1}{A_{i,j}})^{-1};$$

median(A) — возвращает медиану (средний элемент) массива **A**;

mode(A) — возвращает значение наибольшего элемента массива, если он явно есть, в противном случае дает сигнал ошибки;

stdev(A) — задает стандартное отклонение элементов массива **A** — $\sqrt{\text{var}(A)}$;

Stdev(A) — задает выборочное стандартное отклонение элементов массива A — $\sqrt{Var(A)}$;

var(A) — возвращает так называемую смещенную оценку дисперсии (вариацию) для элементов массива **A**:

$$\text{var} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left| A_{i,j} - mean(A) \right|^2;$$

Var(A) — возвращает несмещенную оценку дисперсии для элементов массива **A** с иной, чем у функции var(A), нормировкой:

$$\text{var} = \frac{1}{mn - 1} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left| A_{i,j} - mean(A) \right|^2;$$

kurt(A) — возвращает значение эксцесса (остроты кривой распределения):

$$kurt = \frac{mn(mn+1)}{(mn-1)(mn-2)(mn-3)} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left(\frac{A_{i,j} - mean(A)}{Stdev(A)} \right)^4 - \frac{3(mn-1)^2}{(mn-2)(mn-2)}.$$

Когда речь идет о некоторых зависимостях, данные обычно представляются двумя и более массивами. Степенью связи зависимостей является *коэффициент корреляции* corr(A, B) — коэффициент корреляции Пирсона двух массивов **A** и **B** (m×n элементов). Если коэффициент корреляции близок к 0, то данные (зависимости) не согласованы, а при близости его к 1 данные согласованы.

Применение этих функций системы Mathcad в моделях системы VisSim обеспечивает возможность достаточно полноценного статистического моделирования или просто статистической обработки информации, получаемой в ходе моделирования. Разумеется, множество этих и других статических функций есть и в более мощной системе MATLAB, с которой VisSim также интегрируется.

1.7.5. 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 $t = 0, 1, 2, \ldots$), приведем несколько распространенных моделей дискретных объектов для временной области, учитывающих действие шума наблюдения [30, 31].

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

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

где

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

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

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

или в развернутом виде:

$$y(t) + a_1 y(t-1) + \ldots + a_{na} y(t-n) = b_1 u(t) + b_2 u(t-1) + \ldots + b_{nb} y(t-m) + e(t).$$

Здесь и ниже $e(t)$ — дискретный белый шум, $B(z) = b_1 + b_2 z^{-1} + \ldots + b_{nb} z^{-nc}$.

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

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

$$y(t) = \frac{B(z)}{F(z)} u(t - nk) + e(t), \text{ где } F(z) = 1 + f_1 z^{-1} + f_2 z^{-2} + \ldots + f_{nz} z^{-nf}.$$

- 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 $D(z) = 1 + d_1z^{-1} + d_2z^{-2} + ... + d_{nd}z^{-nd}$.

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):

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

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

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

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

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

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

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

1.8. Методы оценивания параметров

В общем случае *оценивание* параметров модели заданной структуры проводится путем минимизации выбранного критерия качества модели (чаще всего среднего квадрата рассогласования выходов объекта и его постулируемой модели). Рассмотрим несколько возможных подходов к такому оцениванию [31].

1.8.1. Оценивание параметрических моделей

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

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

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

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

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

Из данных выражений следует, что

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

При выборе в качестве критерия (функции потерь) величины

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

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

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

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

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

1.8.2. Оценивание импульсной характеристики

Предположим, что входной сигнал исследуемого (дискретного) объекта имеет нулевое математическое ожидание и является дискретным белым шумом, т. е. имеет автокорреляционную функцию (АКФ):

$$R_u(\tau) = E\{u(t + \tau)u(t)\} = \begin{cases} \lambda, \text{ если } \tau = 0 \\ 0, \text{ в противоположнос случае} \end{cases},$$

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

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

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

получим

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

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

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

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

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

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

Отсюда приходим к оценке ИХ по экспериментальным данным:

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

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

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

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

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

Заметим, что рассмотренная процедура относится к методам корреляционного анализа.

1.8.3. Оценивание спектров и частотных характеристик

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

Для приведенной модели связи между спектрами сигналов описываются известными соотношениями:

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

где $S_{yu}(\omega)$ — взаимная спектральная плотность сигналов $y(t)$ и $u(t)$, которые можно использовать для нахождения оценок комплексного коэффициента передачи объекта $W(e^{i\omega T})$ и спектра шума $S_v(\omega)$. Методика оценивания и нужные для этого соотношения приведены в [31, 40].

В целом представленный в этой главе теоретический материал достаточен для понимания сути реализации описанных в данной книге моделей и примеров моделирования. Разумеется, он не содержит множества деталей, которые присущи каждой модели и интересны лишь тем читателям, которые работают в определенных сферах математического моделирования. Знакомство с такими деталями — прерогатива специальной литературы, например [1—12, 40].

Глава 2
Система визуального моделирования VisSim

Эта глава призвана дать начальное представление о системе математического моделирования VisSim 4.5 и основах работы с этой системой. В ней дается также материал по пользовательскому интерфейсу ряда версий системы VisSim 3 и VisSim 5.

2.1. Место системы VisSim

В последние годы был создан ряд новых программных продуктов для математического моделирования различных явлений, систем и устройств с помощью персональных компьютеров. Заслуженную популярность приобрели специализированные системы схемотехнического моделирования P-Spice, Design LAB, MicroCAP, Electronics Workbench [17—19] и др. Подобные системы созданы и для физического моделирования в физике, химии, механике и проектировании механизмов и др. Однако по-прежнему особый интерес вызывают универсальные системы математического моделирования, способные к интеграции с системами компьютерной математики. Среди наиболее известных из них стоит отметить систему Simulink, интегрированную с мощной матричной математической системой MATLAB [26—29].

Хотя Simulink является важным шагом к наглядному блочному моделированию динамических систем, математическая сторона решения задач моделирования в этой программной системе все еще недостаточно открыта для пользователя. Библиотека компонентов Simulink в основном состоит из наборов специализированных блоков, относящихся к различным предметным областям, например к построению моделей сигналов, механических и электромеханических, электроэнергетических и силовых электронных устройств и т. д. К сожалению, недостатком системы Simulnk+MATLAB, мешающим широкому применению системы в вузах и университетах, является ее крайняя громоздкость.

Ниже в качестве основной описана система математического моделирования VisSim версии 4.5, созданная фирмой Visual Solutions Incorporating. Название системы происходит от слов Visual Simulation — визуальная симуляция. У нас слово «симуляция» давно приобрело нарицательный оттенок, поэтому мы будем использовать более принятый у нас термин «визуально-ориентированное блочное имитационное моделирование» или даже более простой — «визуальное моделирование».

Отличительная особенность системы VisSim — ее явная ориентация на открытое математическое моделирование. Набор блоков для моделей в этой системе резко сокращен (в сравнении с Simulink) и большая часть блоков ориентирована на реализацию именно математических и логических операций. Это позволяет создавать математически прозрачные модели с любыми описывающими их зависимостями, в том числе реализованными в системах компьютерной математики, например таких известных и получивших массовое признание, как Mathcad и MATLAB.

Данная версия VisSim поставляется и интегрируется с массовыми системами компьютерной математики Mathcad 2000/2001/2001i/11 [4], благодаря чему она вполне доступна и имеется на ряде выпущенных в России CD-ROM. Интеграция VisSim 4.5 с математическими системами Mathcad и MATLAB открывает новые возможности выполнения самых серьезных вычислений, причем как в ходе моделирования, так и при обработке его результатов.

Разработчиком VisSim рекламируется как самая быстрая система моделирования. И действительно, время моделирования систем и устройств в VisSim от 3 до 8 раз меньше, чем у других систем моделирования. Это большое достоинства системы, нередко оправдывающее переход к ее применению.

2.2. Назначение и состав системы VisSim 4.5

2.2.1. Назначение системы

Система VisSim (для определенности наиболее распространенная версия VisSim 4.5) предназначена для решения задач математического моделирования, относящихся к следующим классам:

- линейные системы;
- нелинейные системы;
- непрерывные во времени системы;
- дискретные во времени системы;
- системы с изменяемыми во времени параметрами;
- гибридные системы;
- многоцелевые и многокомпонентные системы;
- одновходовые и одновыходные (одномерные) системы SISO;
- многовходовые и многовыходовые (многомерные) системы MIMO;
- гибридные системы.

Система VisSim не имеет явной ориентации на какой-то класс моделирования. Это универсальная система, допускающая достаточно простое расширение и обеспечивающая легкую адаптацию под решение тех или иных конкретных задач пользователя. Тем не менее можно считать, что наиболее удобна данная система для решения задач в области автоматического регулирования и управления, а также при моделировании различных физических, химических, экономических и прочих явлений и систем.

2.2.2. Аппаратные средства для работы с системой

VisSim 4.5 может работать в среде операционных систем Windows 95/98/NT 4+. Была проверена и работоспособность системы в среде Windows 2000/XP. Никаких отклонений в работе системы выявлено не было. VisSim требует весьма скромных аппаратных ресурсов — ПК должен иметь оперативную память с минимальным объемом 4 Мб и объем свободного пространства на жестком диске около 30 Мб. Обязательно наличие математического сопроцессора, поскольку в ходе моделирования широко используется 64-битный формат чисел с плавающей точкой, обеспечивающий очень малые вычислительные погрешности.

2.2.3. Состав библиотеки блоков

Для построения моделей в системе VisSim используются блоки, которые хранятся в библиотеке блоков и могут браться из нее, переноситься в окно модели и соединяться друг с другом.

Библиотека блоков, представленная в позиции Blocks (Блоки) меню и инструментальными панелями, содержит следующие «тома»:

- Animation — блоки создания анимационных клипов;
- Annotation — блоки создания комментариев и определения переменных;
- Arithmetic — блоки арифметических и близких к ним операций;
- Boolean — блоки задания операций Булевой алгебры;
- DDE — блоки интерфейса;
- Integration — блоки задания операций интегрирования;
- Linear Systems — блоки задания параметров пространства состояний линейных систем и их передаточных функций;
- MATLAB Interface — блоки интеграции с матричной системой MATLAB;
- Matrix Operations — блоки задания матричных операций;
- Nonlinear — блоки нелинейных операций и создания нелинейных систем;
- Optimization — блоки задания операций оптимизации;
- Random Generator — блоки генерации случайных чисел;
- Real Time — блоки для систем реального времени;
- Signal Consumer — блоки регистрации, индикации и построения графиков сигналов;
- Signal Producer — блоки создания сигналов;
- Time Delay — блоки создания временной задержки;
- Transcendental — блоки задания трансцендентных математических функций;
- General — функции общего характера.

Названия блоков, имеющихся в этих разделах библиотеки, представлены ниже:

- Animation: animate; lineDraw.
- Annotation: bezel; comment; date; index; label; scalarToVec; variable; vecToScalar; wirePositioner.

- Arithmetic: 1/X; –X; *; /; abs; convert; gain; pow; sign;summingJunction; unitConversion.
- Boolean: >; <; >=; <=; ==; !=; and ; not; or; xor.
- DDE: DDE; DDEreceive; DDEsend.
- Integration: integrator; limitedIntegrator; resetIntegrator.
- Linear Systems: stateSpace; transferFunction.
- MATLAB Interface: MatLab Expression; MatLab Read Variable; MatLab Write Variable.
- Matrix Operations: buffer; dotProduct; fft; ifft; invert; multiply;psd; transpose; vsum.
- Nonlinear: case; crossDetect; deadband; init; limit; map; max; merge; min; quantize; relay;sampleHold.
- Optimization: constraint; cost; globalConstraint; parameterUnknown; unknown.
- Random Generator: gaussian; uniform; PRBS.
- Real Time: rt-DataIn; rt-DataOut; ActiveXread; ActiveXwrite.
- Signal Consumer: display; error; export; histogram; light; meter; plot; stop; stripChart.
- Signal Producer: button; const; dialogConstant; import; parabola; pulseTrain; ramp; realTime; sinusoid; slider; step.
- Time Delay: timeDelay; unitDelay.
- Transcendental: acos; asin; atan2; bessel; cos; cosh; exp; ln; log10; sin; sinh; sqrt; tan; tanh.
- General: embed*; expression; OLEitem; userFunction*.

* Эти функции в VisSim PE (Personal Edition) не используются.

Набор блоков системы может быть расширен установкой пакетов расширения системы. Пока их намного меньше, чем у системы Simulink. Один из пакетов расширения VisSim/Com содержит, к примеру, около 170 новых блоков для построения моделей коммуникационных устройств. Полное описание упомянутых выше блоков имеется в главе 3.

2.2.4. Основные обозначения в блоках

При описании блоков и окон установки их параметров полезно придерживаться определенных обозначений. В VisSim приняты следующие основные обозначения:

A — амплитуда;
e — указатель порядка чисел в научной нотации;
dt — производная;
lb (Lower bound) — нижний предел;
mod — модуль;
s — оператор Лапласа;
t — время;
ub (Uper bound) — верхний предел;
ω — круговая частота ($\omega = 2\pi f$, где f — частота в герцах);
x — входной сигнал;

y — выходной сигнал;

$ — первый символ системных переменных.

Некоторые блоки имеют несколько входов и (или) выходов. Они обозначаются как x_1, x_2, ..., x_n или x_1, x_2, ..., x_n для входов и y_1, y_2, ..., y_n или y_1, y_2, ..., y_n для выходов. Входы и выходы обозначаются цветными треугольниками, острие которых указывает на направление подачи сигнала (для входных блоков внутрь блока, для выходных — из блока).

Правилом хорошего тона является введение текстовых меток label — как в окнах свойств блоков, так и с помощью блока label. Практика, однако, показывает, что подобно тому, как программисты забывают о комментариях в своих программах, так и разработчики моделей забывают маркировать метками блоки в них. По умолчанию метки не задаются или задаются англоязычными словами. Но никаких препятствий вводить русскоязычные сетки нет.

Обзор каждого блока системы с примером или с примерами применения дан в следующей главе. Подробные данные о блоках, их обозначения и изменяемые параметры можно найти в справке по системе VisSim и в ее техническом описании, представленном файлами формата PDF. Однако для их прочтения надо владеть английским языком.

2.2.5. Вьюверы системы

Система VisSim имеет интегрированный с ней вьювер — средство задания и просмотра диаграмм моделей и результатов моделирования. Вьювер выпускается также отдельно как бесплатная программа, позволяющая просматривать модели, но не модифицировать и записывать их. Вьювер позволяет в левой части экрана просматривать древообразную структуру модели, а в правой — графическое представление модели и результатов ее работы. Границу между ними можно перемещать мышью.

Для просмотра результатов моделирования предусмотрено подключение к нужным точкам модели необходимых сигналов и констант, а также регистраторов сигналов. В отличие от системы Simulink регистраторы-графопостроители (виртуальные осциллографы) имеют окна, которые располагаются прямо в окне модели. Для простых, в частности учебных, моделей это довольно удобное решение. Окна регистраторов имеют типичный вид окон Windows-приложений, их можно перемещать, сворачивать и разворачивать. Возможно их размещение как в основной модели, так и во входящих нее субмоделях.

2.3. Первый пример применения системы VisSim

2.3.1. Загрузка примера — модели резонансного инвертора

Знакомство с VisSim проще всего начать с исполнения одного из многочисленных примеров, поставляемых вместе с системой и описанных (с рядом оригинальных примеров) в последующих главах. Возьмем пример моделирования резонансного инвертора (файл Inverter.vsm). Этот инвертор преобразует постоянное входное напряжение в почти синусоидальное переменное напря-

жение на нагрузке. Подобные инверторы широко применяются в так называемых вторичных источниках электропитания различных устройств (первичными называют сеть переменного тока или химические источники энергии).

Запустив систему VisSim с помощью команды Open (Открыть) меню File (Файл) можно открыть стандартное окно загрузки файлов (рис. 2.1). В представленной в нем файловой системе надо открыть и загрузить файл Invertor.

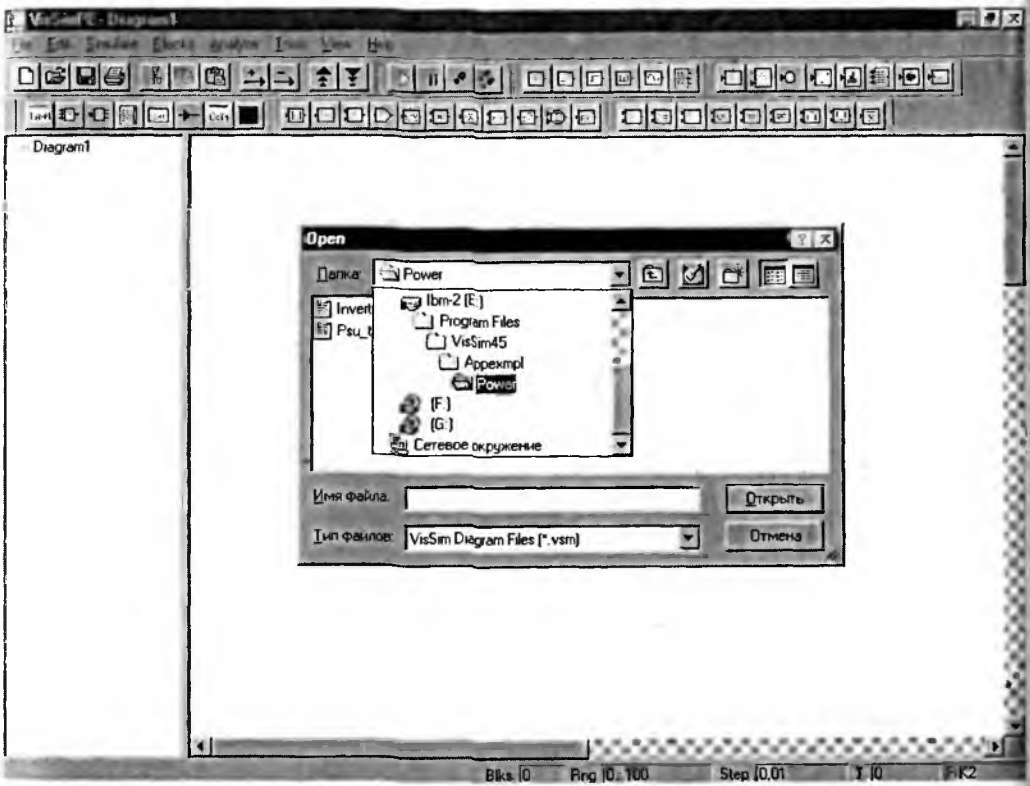


Рис. 2.1. Окно системы VisSim 4.5 и окно загрузки файлов моделей

Как видно из рис. 2.1, поначалу большое окно модели пусто, а в левом окне имеется корневая диаграмма Diagram 1. В дальнейшем подобная диаграмма или модель будет называться *основной диаграммой*, а входящие в нее модели и блоки более низкого уровня мы будем называть *субмоделями* (подмоделями) и *субблоками* (подблоками).

2.3.2. Просмотр структуры модели

VisSim обеспечивает создание моделей с включенными в них подмоделями (субблоками). Это и поясняет пример моделирования резонансного инвертора, показанный на рис. 2.2. Здесь отчетливо видны особенности интерфейса системы VisSim — окно древообразной структуры модели и окно самой модели. Как и у всех приложений под Windows, сверху окна системы имеется титульная строка, строка меню и панели инструментов. Снизу имеется строка статуса системы.

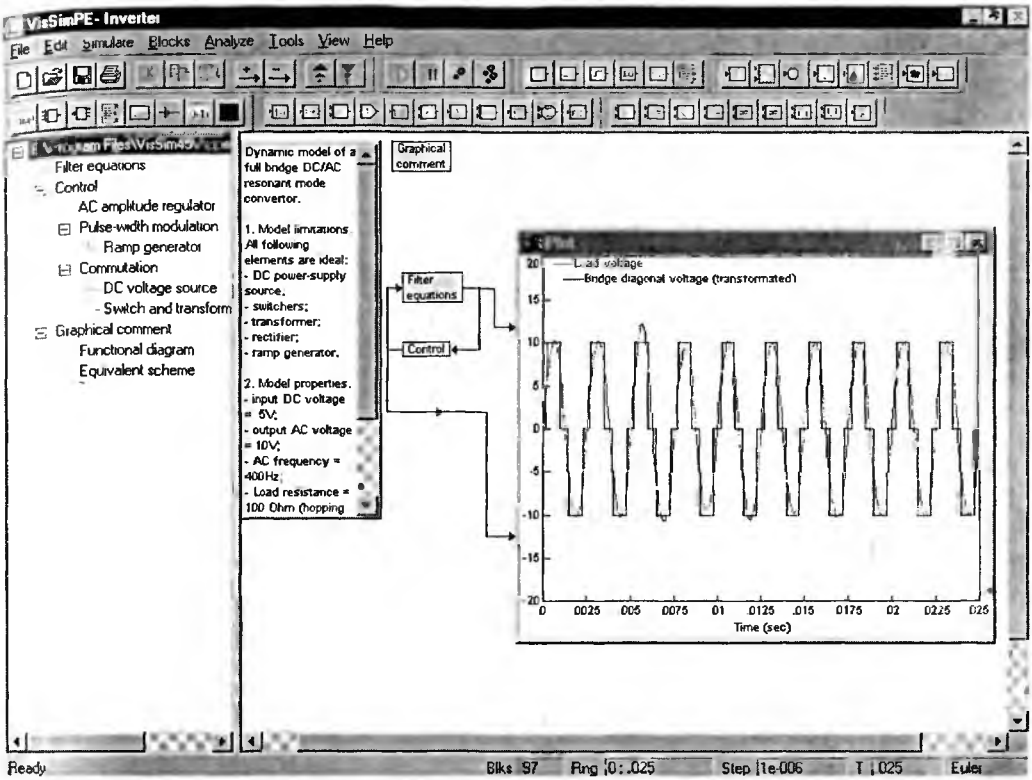


Рис. 2.2. Модель резонансного инвертора

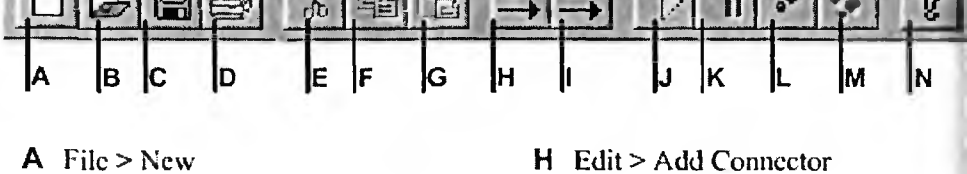
В нашем случае модель сведена всего к двум замкнутым в кольцо субблокам — уравнениям фильтра Filter Equations и контроля Control. Есть также подблок графических комментариев Graphical Comment. Все субблоки имеют синий цвет фона. Кстати, в системе предусмотрено изменение окраски блоков по ходу моделирования, но оно не всегда заметно из-за его высокой скорости.

Модель, сведенная к простой структурной схеме из ряда достаточно сложных моделей, принято именовать *макромоделью*, а моделирование с ее помощью тех или иных устройств и систем называют *макромоделированием*. Мы не будем специально акцентировать внимание на этом, поскольку провести четкую грань между моделированием и макромоделированием практически невозможно. О любой модели можно сказать, что она может быть расчленена на более мелкие блоки, подобно тому как реальное устройство можно делить на более мелкие части, видимо, вплоть до атомов.

При наведении курсора мыши на субблок курсор превращается в крестик со стрелками. Теперь при нажатой левой клавиши мыши субблок можно перемещать мышью по полю окна модели, а двойным щелчком мыши можно открыть субблок и рассмотреть его графическое представление в виде подмодели. Примеры этого мы приведем ниже.

Начинающего пользователя может напугать обилие кнопок на панелях инструментов. Однако при внимательном рассмотрении оказывается, что

лишь небольшое число кнопок служит для управления системой. Они пр
ставлены на рис. 2.3. Эти кнопки дублируют файловые операции, операц
редактирования и управления моделированием. Остальные панели инстр
ментов содержат кнопки вывода блоков, дублируют команды позиции Blo
меню и по виду соответствуют изображению того или иного блока в диагра
ме модели. Название каждой кнопки (блока) выводится в виде всплываю
подсказки при наведении на кнопку курсора мыши.



A File > New

B File > Open

C File > Save

D File > Print

E Edit > Cut

F Edit > Copy

G Edit > Paste

H Edit > Add Connector

I Edit > Remove Connector

J Simulate > Go

K Simulate > Stop

L Simulate > Single Step

M Simulate > Continue

N Help

Рис. 2.3. Панели инструментов с кнопками общего назначения

Строго говоря, надо различать понятие модели и диаграммы модели, т. е.
графического представления. Но подобно тому как в радиотехнике и электр
нике слово «схема» давно отождествляется с реальными устройствами (так и
ворят «схема работает» или «схема не работает»), так и диаграмма модели в V
Sim нередко означает именно модель, а не просто ее графическое изображени

Для пуска моделирования достаточно нажать кнопку J. В окне графич
ского контроля при этом появляются результаты моделирования в виде вр
менных зависимостей напряжения в нагрузке (оно почти синусоидальное)
на выходе мостового преобразователя (см. рис. 2.2).

Модель может сопровождаться различной документацией. Например, э
может быть окно с кратким текстовым описанием модели. Такое окно пока
но в окне модели слева (см. рис. 2.2). Можно также ввести и затем просма
ривать графические комментарии в виде различных схем и диаграмм. В пр
веденном примере они выводятся активизацией блока Graphical Comment.
выводит два блока — Funcional Diagramm (Функциональная схема) и Eqival
sheme (Эквивалентная схема). На рис. 2.4 представлена функциональная сх
ма инвертора. Точнее говоря, это почти полная принципиальная схе
устройства с выделением на ней функциональных узлов блок-схемы.

Эту схему можно привести к более простой эквивалентной схеме, пок
занной на рис. 2.5, что соответствует концепции макромоделирования систе

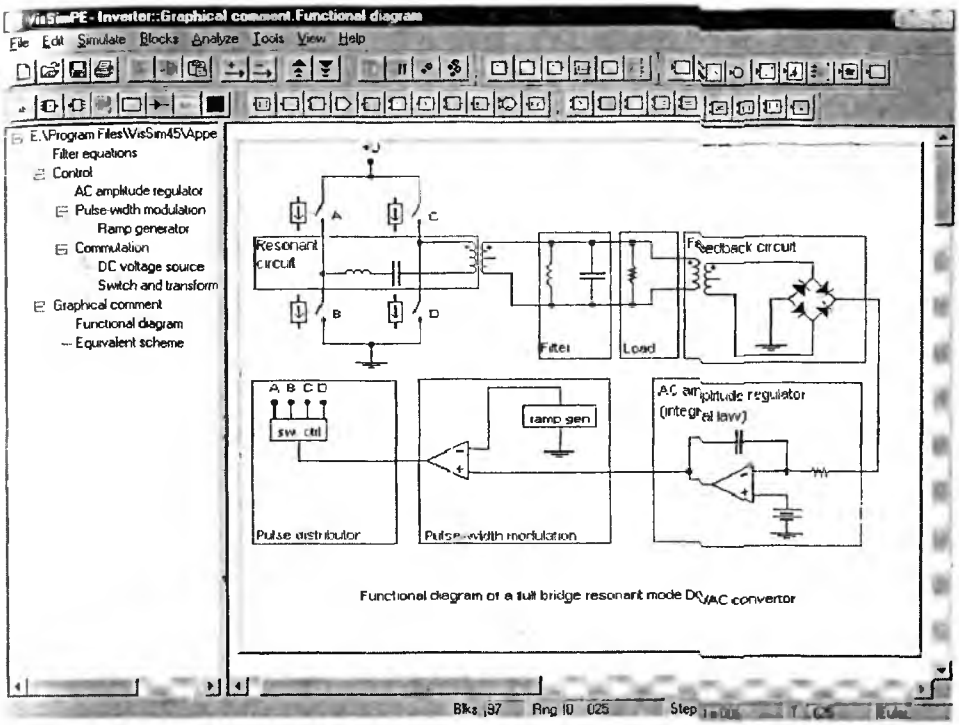


Рис. 2.4. Функциональная схема инвертора

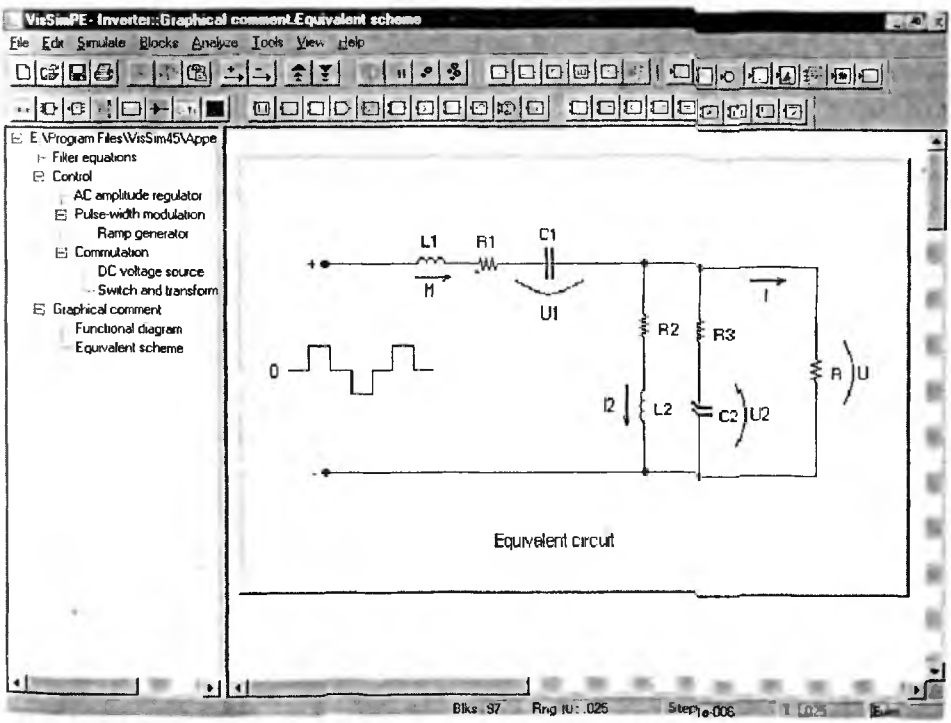


Рис. 2.5. Эквивалентная схема инвертора

Макромодель в данном случае представляет собой типичный резонансны фильтр, на вход которого подается импульсный сигнал от преобразовате постоянного напряжения в импульсное напряжение. Фильтрация этого сигна ла резонансным фильтром и порождает почти синусоидальное напряжение нагрузке. Подобный метод преобразования широко используется в источни ках вторичного электропитания компьютерной и связной аппаратуры, поск льку обеспечивает малый уровень импульсных помех. Исходное (входное) п стоянное напряжение обычно создается подзаряжаемым аккумулятором бол шой емкости.

Главный субблок модели Filter Equations представляет собой набор бл ков, задающих и решающих уравнения для эквивалентной схемы рис. 2. Этот субблок представлен на рис. 2.6. Блок создается из отдельных блоков м дели, каждый из которых имеет свое окно для установки параметр (свойств — properties). Одно из таких окон для интегратора 1/S также пре ставлено на рис. 2.6.

Аналогичным описанному выше образом можно легко просмотреть п строение субблока Control и ознакомиться с деталями реализации источни двухполярного импульсного напряжения, подаваемого на вход фильтра. Соо ветствующие диаграммы находятся в разделе Commutation. На рис. 2.7 пре ставлен субблок амплитудного модулятора AC amplitude regulator. В раздел Pulse-width modulator и Communication можно найти и подмодели других ко понентов субблока Communication.

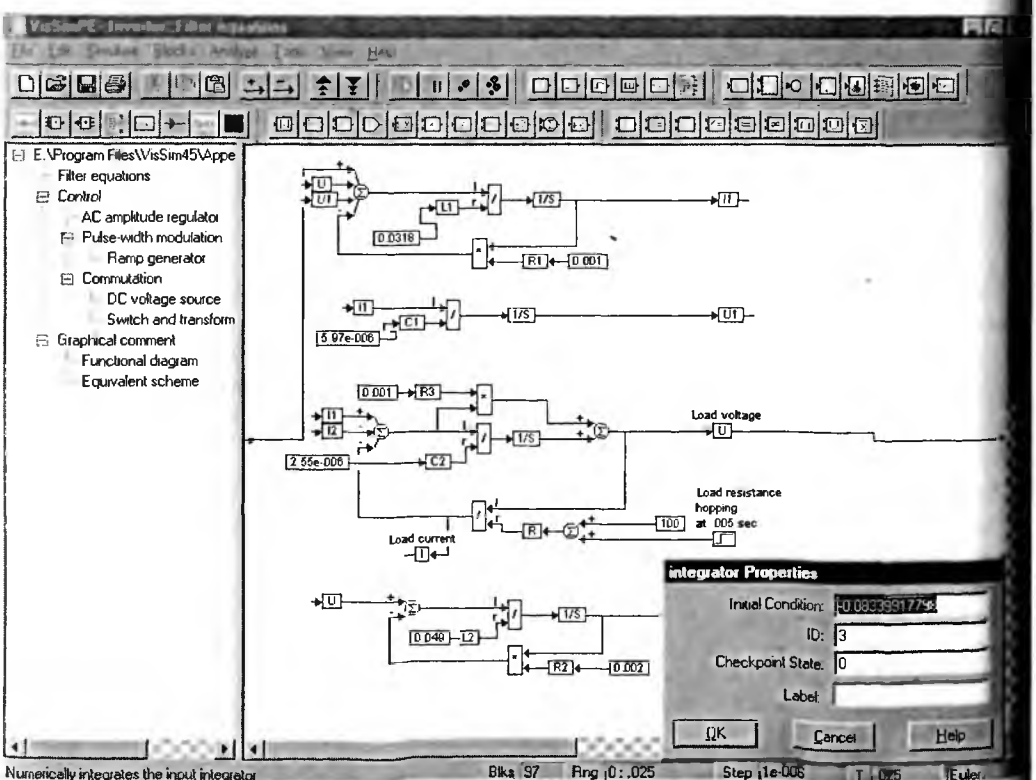


Рис. 2.6. Субблок задания фильтра Filter Equations

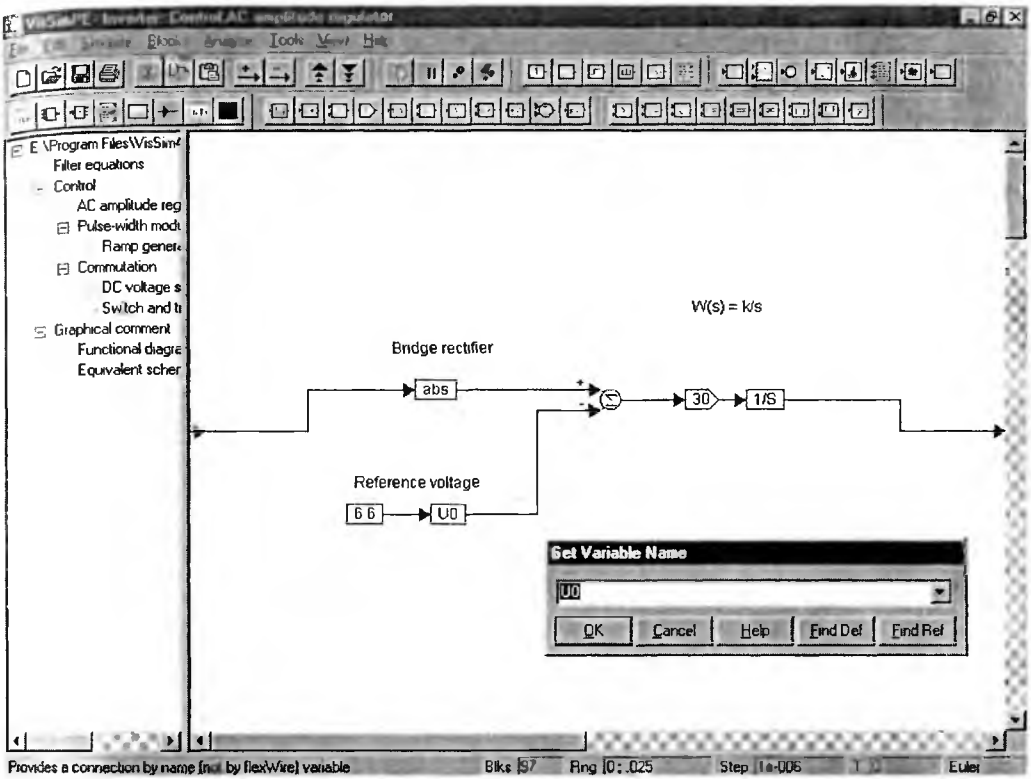


Рис. 2.7. Субблок контроля Control

Теперь вернемся к моделированию. После запуска модели кнопкой J (рис. 2.2, кнопка имеет вид зеленого треугольника) начинается процесс моделирования и строятся заданные временные диаграммы напряжений на входе и выходе фильтра. Рисунок 2.8 показывает эти диаграммы после раскрытия окна временных диаграмм и удаления (перемещением мышью раздела деления экрана надвое) древообразной диаграммы модели.

Для получения правильных результатов моделирования необходима некоторая настройка блоков модели, установка параметров моделирования и форматирование графиков. Как отмечалось, настройка блоков осуществляется с помощью окна свойств (параметров) каждого блока. В позиции меню Simulate можно найти установки параметров моделирования, в частности выбор метода решения дифференциальных уравнений, описывающих моделируемую систему, начальных значений, шага интегрирования и др. Средства реализации ряда методов анализа можно найти в позиции Analyze меню.

Для форматирования графиков VisSim достаточно щелкнуть по ним мышью. Появится окно форматирования установки параметров графопостроителя Plot, показанное на рис. 2.8 справа. Это окно имеет 5 вкладок, установки на которых достаточно очевидны и будут более подробно описаны в дальнейшем. Следует отметить высокое качество графиков и оформления их осей и комментарий.

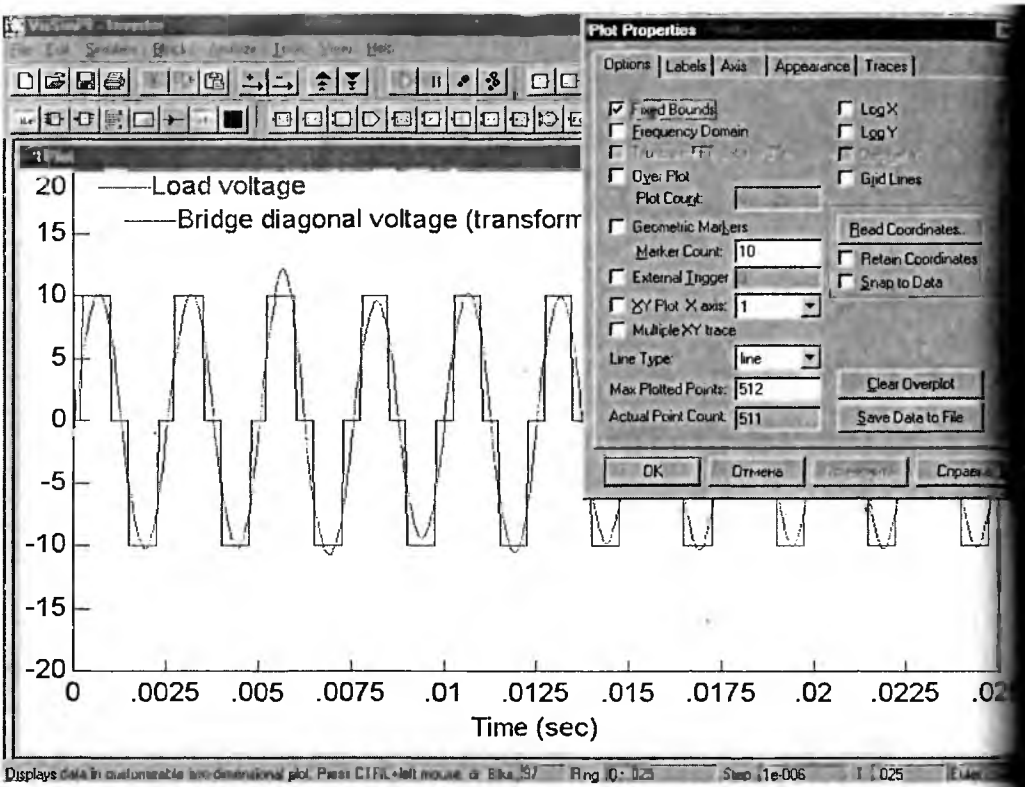


Рис. 2.8. Временные диаграмм на входе и выходе фильтра при раскрытом графопостроителя Plot

Иногда моделирование сложных систем и устройств идет довольно ме ленно, занимая минуты, а то и часы. Остановить моделирование, если его р зультаты вас перестали интересовать или просто нет времени на досмотр е результатов, можно нажатием кнопки рядом — с двумя жирными вертикал ными линиями. Остановка может произойти не сразу, а по завершении тек щего шага моделирования. Признаком остановки является смена ставшего с рым треугольника на кнопке запуска моделирования на зеленый треугольни

2.3.3. О возможностях VisSim

Из приведенных примеров хорошо видно, что система VisSim имеет удо ные и наглядные средства для построения блочных математических моделей встроенными в них субмоделями на основе математически ориентированны блоков. Число блоков в библиотеке системы около 100, что немного. Но п льзователь может легко подготовить свои модели и записать их в виде файло Ряд таких файлов поставляется в стандартной поставке системы. Наконец п полнить набор блоков можно с помощью пакетов расширения.

Технические возможности системы VisSim 4.5 PE предусматривают при менение до по 500 Мб памяти под блоки, точки графических окон, точки ин теграторов и т. д. Это говорит о возможности моделирования весьма сложны

систем. При численных расчетах поддерживается 64-битовый стандарт IEEE для чисел с плавающей точкой. Поддерживаются форматы файлов ASCII, .M, .MAT, .WAV, что обеспечивает легкую стыковку системы с различными математическими системами, прежде всего с Mathcad и MATLAB. Все это свидетельствует о достаточно высоких перспективах применения системы VisSim в практике моделирования различных явлений, систем и устройств.

2.4. Об интеграции VisSim с системой Mathcad

2.4.1. Основные возможности интеграции

Интеграция системы VisSim с другими программными средствами расширяет круг задач моделирования. Особенно важна такая интеграция с современными СКМ. VisSim имеет специальные средства для интеграции с матричной системой MATLAB:

- импорт файлов пространства состояний MATLAB;
- динамическое изменение переменных в блоках VisSim и MATLAB;
- исполнение в среде VisSim выражений, входящих в блоки.

Подробно они будут рассмотрены в главе 6. Как уже отмечалось, VisSim может работать и с системами компьютерной математики Mathcad. Возможность работы VisSim с системой MATLAB на первый взгляд выглядит не слишком убедительной, поскольку MATLAB поставляется вместе со своим пакетом блочного имитационного моделирования Simulink. Так что особой необходимости применять VisSim при использовании системы MATLAB нет.

А вот работу с популярной и дешевой системой Mathcad можно, несомненно, отнести к весьма важному достоинству VisSim, кстати отсутствующему у Simulink. Оно обеспечивает поистине неисчерпаемые возможности в создании моделей, реализующих самые эффективные численные и иные методы вычислений, причем как самые простые, так и самые сложные. Возможно практически полное использование средств системы Mathcad в блоках, располагаемых в VisSim моделях. Это делает такие модели одновременно весьма наглядными и мощными (в смысле решения множества разнообразных задач моделирования).

2.4.2. Внедрение блока Mathcad в модель системы VisSim

Интеграция VisSim с Mathcad осуществляется путем применения специальных блоков — объектов Mathcad Object. Для их создания внутри модели надо открыть список Insert Mathcad Object (Вставить Mathcad-объект) в позиции Tools (Инструменты) меню. В списке есть две команды:

1) New (Новый) — вставка нового Mathcad-объекта;

2) From File... (Из файла) — открытие окна загрузки Marhcad-файлов.

При исполнении команды New запускается система Mathcad, и в окне модели появляется объект системы Mathcad (рис. 2.9). Он имеют вид прямоугольника с широкой заштрихованной рамкой со входами (слева) и выходами (справа). Строка меню VisSim заменяется строкой меню системы Mathcad и в

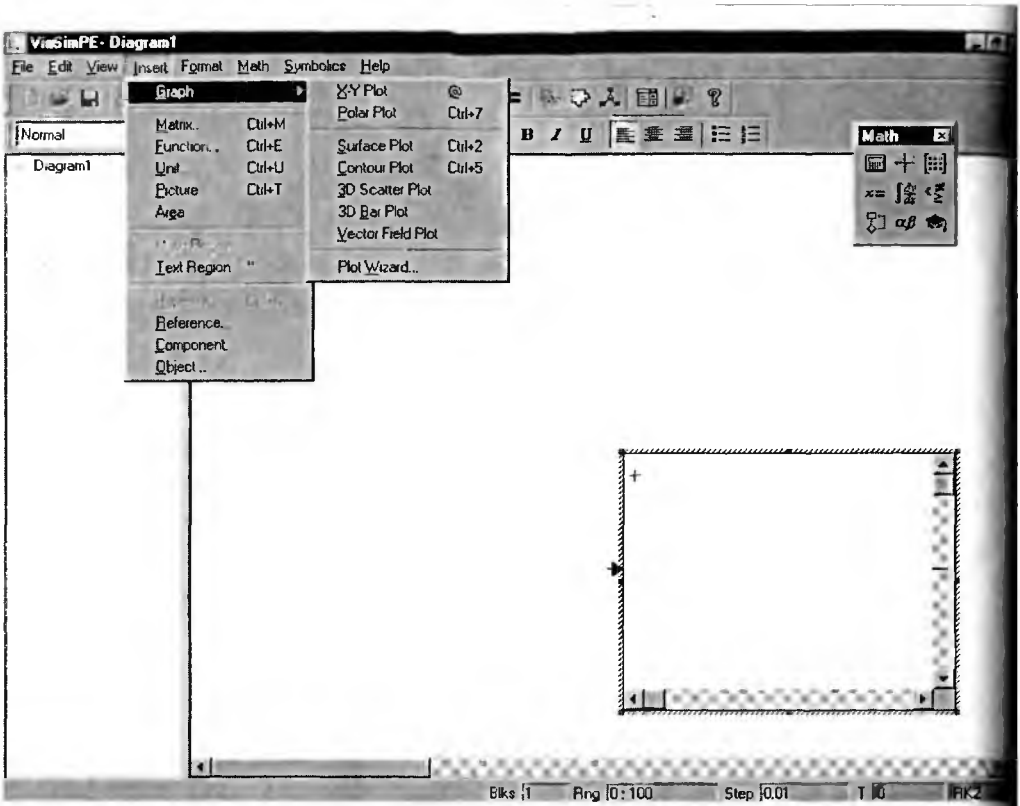


Рис. 2.9. Внедрение окна системы Mathcad в окно модели VisSim

окне модели появляется плавающая панель выбора палитр математическ символов.

Поначалу блок имеет только один вход и один выход (тип SISO). Одна с помощью команды Add Connector в позиции Edit меню можно (перемещ нием объекта-стрелки мышью) добавить вход или выход, т. е. превратить бл Mathcad в блок типа MIMO. Для обеспечения интерфейса связи между Ma cad блоком и моделью VisSim в Mathcad предусмотрено задание интерфе сных переменных in0, in1, in2 и т. д. для входов и out0, out1, out2 и т. д. д выходов.

2.4.3. Простой пример интеграции VisSim с системой Mathcad

Проиллюстрируем возможность интеграции системы VisSim с систем Mathcad на простом примере — есть три константы in0 = 3, in1 = 7 и in2 = 5 надо вычислить средствами системы Mathcad in0 + in1 + in2 и in0 + in1 – i При этом первый результат надо вывести по выходу out0, а второй по выхо out1.

Реализация этого вполне очевидного примера приведена на рис. 2.10. О не нуждается в особых комментариях. Полезно разве что учесть возможнос введения русскоязычных комментариев в Mathcad, блоки. Правила их подг товки можно найти в литературе по системе Mathcad [23].

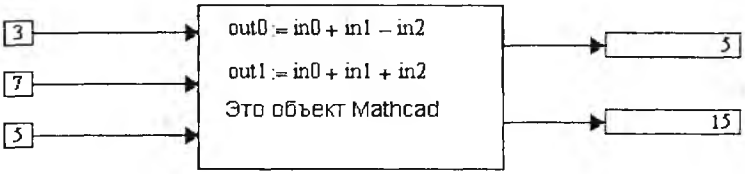


Рис. 2.10. Пример применения простого объекта Mathcad

Заметим, что дважды щелкнув мышью по изображению Mathcad-объекта, можно вернуться в среду Mathcad и начать редактирование Mathcad-объекта. Более подробное описание работы с Mathcad-объектами имеется в главе 6.

2.4.4. Более сложный пример интеграции VisSim с системой Mathcad

Рисунок 2.11 показывает более сложный пример применения Mathcad-блока. Здесь моделируется демпфированный механический осциллятор в виде тележки с массой M прикрепленной к пружинке. Нетрудно заметить, что Mathcad блок может содержать рисунок, имеющий в нашем случае чисто иллюстрационное значение. Если сжать или оттянуть пружину, сместив тележку относительно

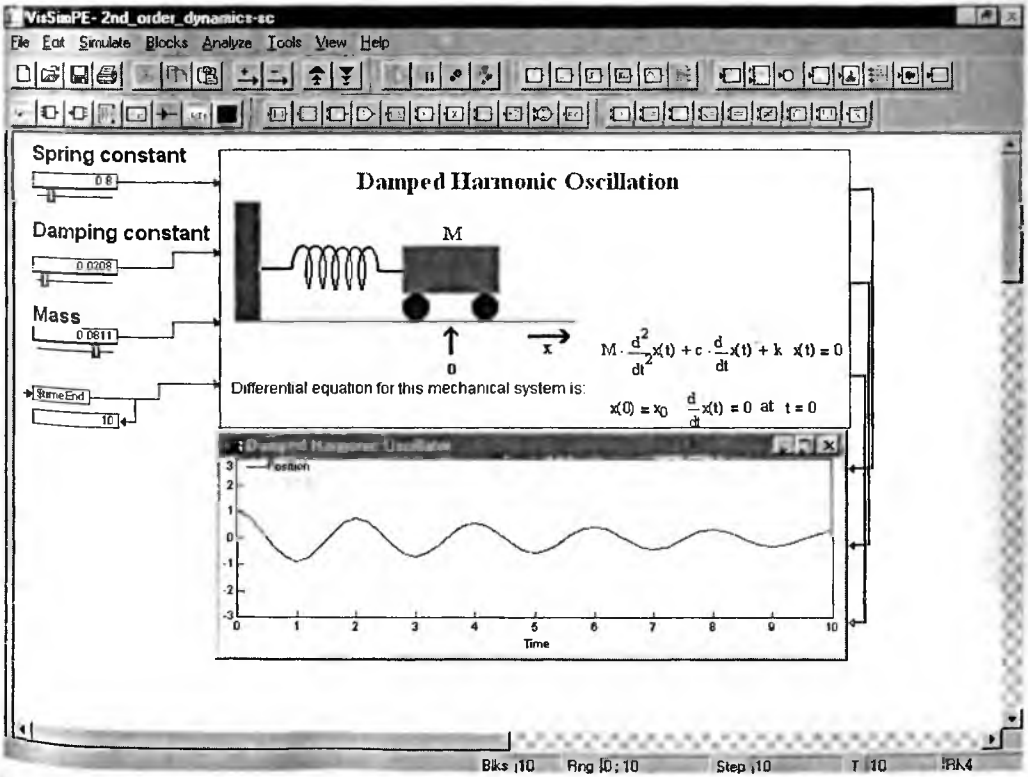


Рис. 2.11. Моделирование демпфированного механического маятника

точки равновесия $x = 0$, то можно наблюдать затухающие колебания тележки, что и фиксирует графопостроитель. В объекте Mathcad задана графическая иллюстрация задачи и приведено решение дифференциального уравнения второго порядка, описывающего затухающие колебания осциллятора.

Как было отмечено, объекты Mathcad можно редактировать не выходя из среды VisSim. Для этого достаточно активизировать объект, установив в него курсор мыши и дважды щелкнув ее левой клавиши. Объект окажется выделенным, и появятся элементы интерфейса системы Mathcad (рис. 2.12). В окно, блока Mathcad теперь можно легко редактировать документ средствами, которые имеет система Mathcad. Чтобы вернуться в среду VisSim, достаточно отвести курсор мыши в сторону от Mathcad-блока и щелкнуть левой клавишей мыши.

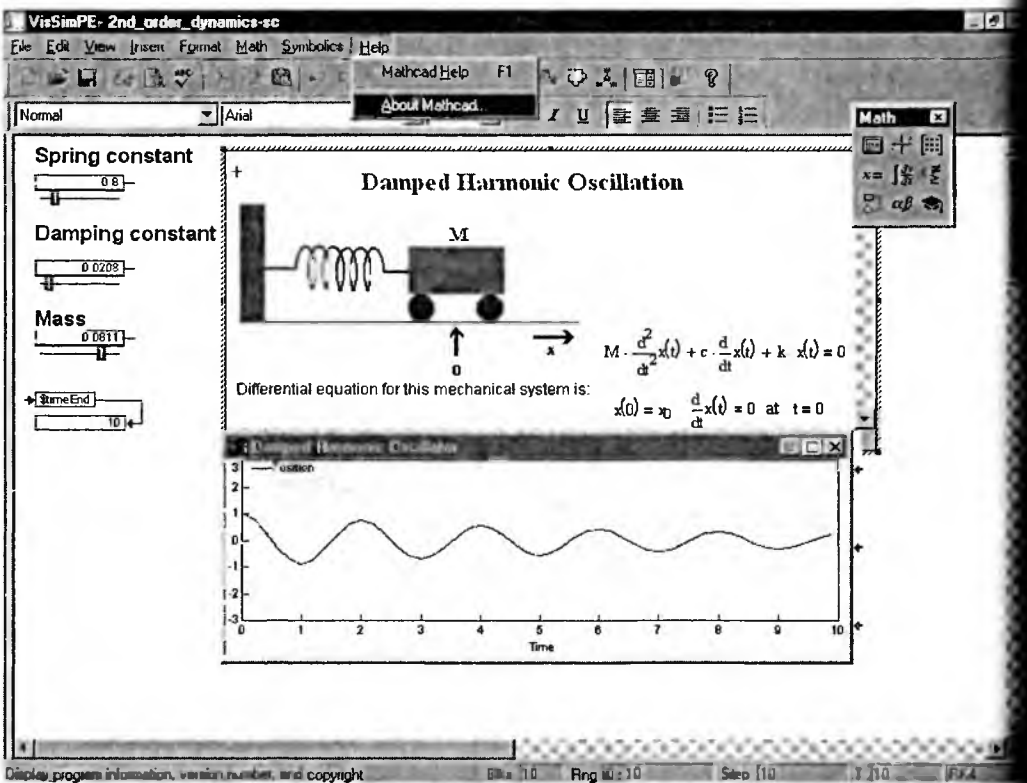


Рис. 2.12. Редактирование объекта — блока Mathcad

Более подробное описание возможностей интеграции системы VisSim с СКМ Mathcad дано в главе 6.

2.5. Пример анимации в системе VisSim

Привлекательности системы VisSim несомненно способствуют простые и наглядные средства *анимации изображений*. Под анимацией подразумевается представление изображения в виде отдельных кадров, у которых изображения

несколько разнятся. Достаточно быстрая смена кадров ведет к появлению эффектов динамики. VisSim имеет обширный набор файлов формата .bmp, которые можно использовать как для создания пиктограмм блоков, так и для подготовки анимационных изображений.

На рис. 2.13 показана модель сердца (демонстрационный файл Physbe vsm) и результаты ее моделирования в виде временных диаграмм.

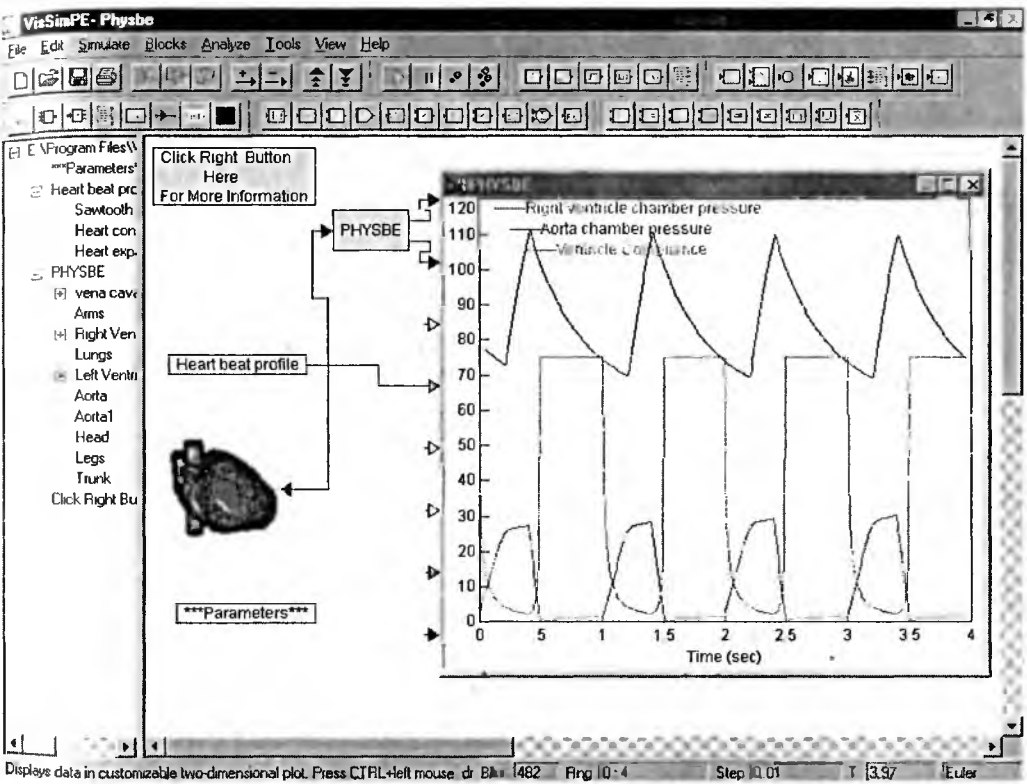


Рис. 2.13. Пример моделирования работы сердца

Во время моделирования можно наблюдать сокращения сердечной мышцы и изменение вида сердца. К сожалению, рис. 2.13 показывает лишь один кадр анимации. Но, пустив этот пример, можно отчетливо наблюдать ритмические сокращения и расширения сердца, наблюдая одновременно осциллограммы его работы. В этом примере использован один из наборов файлов формата .bmp.

2.6. Начало работы с VisSim

2.6.1. Инсталляция и запуск VisSim

Инсталляция (установка) VisSim не имеет каких-либо особенностей и происходит как инсталляция любого приложения под операционную систему Windows. Возможна инсталляция запуском инсталляционного файла (в том

числе полученного из Интернета) или с CD-ROM, в том числе системы Mathcad, поставляемой с VisSim 4.5 PE.

Инсталляция проходит гладко и в особом описании не нуждается. Единственное, что надо отметить, это запрос на интеграцию VisSim с системой MATLAB. На него надо ответить положительно, если вы предполагаете такую возможность. Если нет, что более вероятно, то ответьте на запрос отрицательно.

Для запуска системы VisSim можно выполнить любое из действий:

- активизировать мышью ярлык программы на рабочем столе (рис. 2.14);
- исполнить команду Открыть из контекстного меню правой клавиши мыши (см. также (рис. 2.14);
- выполнить команду Пуск > Программы > VisSim 4.5 PE (или Professional VisSim 4.5) (рис. 2.15).

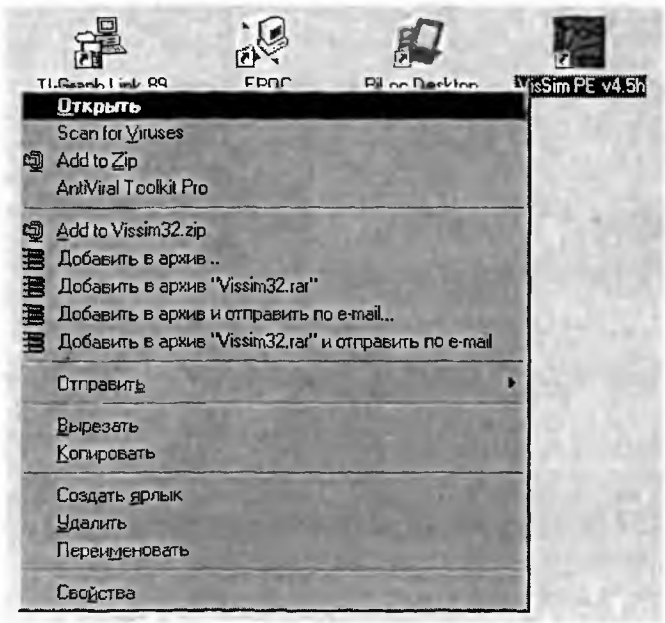


Рис. 2.14. Ярлык системы VisSim на рабочем столе Windows и контекстное меню правой клавиши мыши

Возможен также запуск VisSim с командной строки, что позволяет:

- автоматически открывать определенную блок-схему;
- активировать автозапуск процесса симуляции модели.

Если при запуске в командной строке вводится более чем один параметр, то параметры надо отделять пробелами. Если имя файла блок-схемы включено в список параметров, то оно должно быть определено последним. Могут использоваться следующие аргументы:

- имя_блок-схемы — запуск VisSim и открытие указанной блок-схемы;
- -i [имя_блок-схемы] — запуск VisSim в виде иконки и открытие указанной блок-схемы;

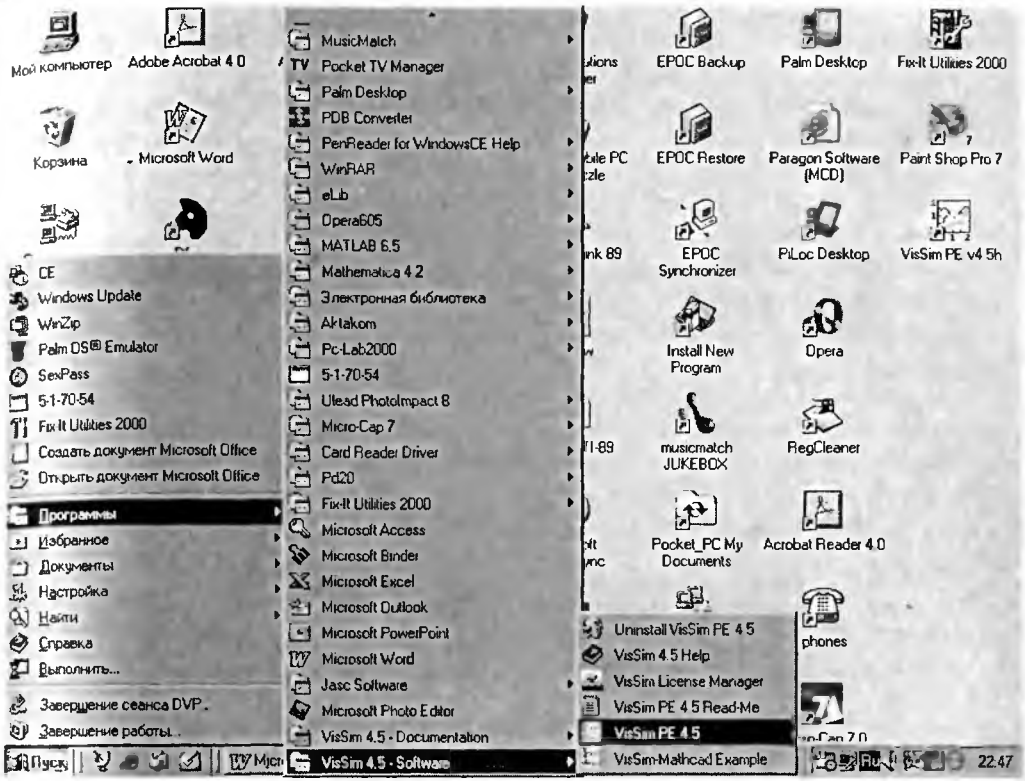


Рис. 2.15. Запуск VisSim из меню программ кнопки Пуск

- -nb [имя_блок-схемы] — запуск VisSim без стартового баннера и открытие указанной блок-схемы;
- -ne — подавление диалогового окна завершения моделирования;
- -r имя_блок-схемы — выполнение симуляции указанной блок-схемы после запуска VisSim;
- -re имя_блок-схемы — выполнение симуляции указанной блок-схемы с последующим выходом из VisSim.

Иногда полезно выполнить нужные установки в окне свойств ярлыка запуска VisSim, доступ к которому имеется в контекстно-зависимом меню правой клавиши мыши (см. рис. 2.14, команда Свойства). Это окно показано на рис. 2.16 с открытой одной из двух вкладок Общие. Это окно информационное, но в нем можно установить атрибуты файла запуска.

На вкладке Ярлык (рис. 2.17) имеется возможность уточнения строки запуска, рабочего каталога и размера окна (с выбором из списка).

В дальнейшем мы не будем пользоваться возможностью настройки запуска системы. Они важны, если VisSim используется в составе некоторого, например презентационного или управляющего, комплекса, который должен начинать работу с уже загруженной моделью или с заведомо установленными настройками.

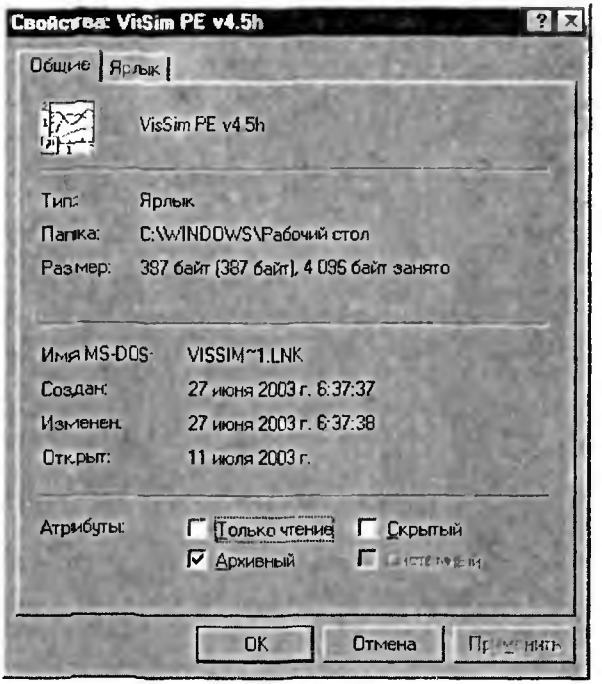


Рис. 2.16. Окно свойств ярлыка запуска системы VisSim с открытой вкладкой Общие

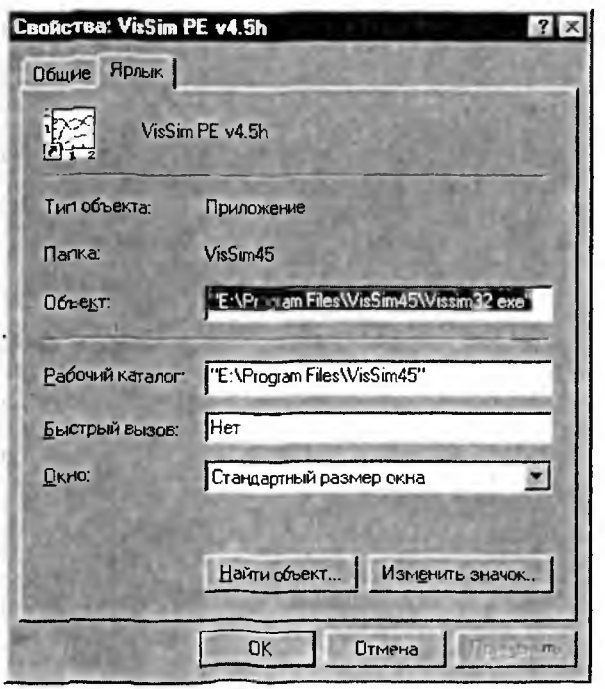


Рис. 2.17. Окно свойств ярлыка запуска системы VisSim с открытой вкладкой Ярлык

2.6.2. Окно VisSim и его настройка

При запуске VisSim появляется окно, подобное тому, что изображено на рис. 2.1 и 2.2. Для получения справки об элементах окна достаточно навести на них курсор мыши и задержать его на пару секунд — появится всплывающая подсказка с наименованием объекта интерфейса на желтом фоне.

Окно системы VisSim содержит следующие главные части:

- титульную строку сверху;
- меню программы;
- две панели с кнопками быстрого управления и вывода блоков;
- окно с графической диаграммой модели (в левой части общего окна);
- большое окно ввода и редактирования модели (в правой части общего окна);
- строку состояния системы (внизу общего окна).

В начале титульной строки имеется кнопка с изображением логотипа системы VisSim. Подобная кнопка есть во всех Windows-приложениях. Ее активизация открывает меню титульной строки — русскоязычное, если применена русифицированная операционная система. Это меню имеет следующие позиции:

- Восстановить — восстановить установки окна;
- Переместить — подготовить окно к перемещению мышью;
- Размер — подготовить окно к изменению размера;
- Свернуть — свернуть окно в бирку, помещаемую в панель задач;
- Развернуть — развернуть окно на весь экран (если оно было меньшего роазмера);
- Закрыть — закрыть окно приложения, т. е. VisSim.

Далее в титульной строке указывается также название программы VisSim и имя файла открытой модели (блок-схемы), например «Diagram1». В правой части строки имеются три стандартные кнопки управления: «Свернуть», «Восстановить/Развернуть» и «Закрыть». Кнопка «Свернуть» превращает окно VisSim в бирку в панели задач, кнопка «Восстановить/Развернуть» меняет размер окна от текущего к максимальному, а и кнопка «Закрыть» закрывает окно VisSim и обеспечивает выход из программы.

2.6.3. Меню системы VisSim и панели инструментов

Интерфейс программы VisSim создан таким образом, что большинство операций может выполняться с помощью *панелей инструментов*. Однако полный доступ ко всем операциям осуществляется из *меню*. Строка меню содержит следующие позиции:

- File (Файл) — файловые операции и операции печати модели;
- Edit (Правка) — операции редактирования модели;
- Simulate (Моделирование или Симуляция) — установка параметров моделирования;
- Blocks (Блоки) — доступ к разделам библиотеки блоков;
- Analyze (Анализ) — доступ к средствам анализа;
- Tools (Инструменты) — доступ к дополнительным инструментальным возможностям;

- View (Вид) — доступ к средствам изменения вида интерфейса;
- Help (Справка) — доступ к справочной системе.

Первая буква каждой позиции меню выделена подчеркиванием. Это зн чит, что эту позицию можно активизировать, нажав выделенную букву одн временно с нажатием клавиши Alt. Этот прием действует и в отношении а тивизации других команд, если в них есть выделенная буква (не обязатель первая). У многих команд меню указываются также горячие клавиши — ко бинации клавиш, при которых можно быстро выполнить ту или иную кома ду без поиска ее на панелях инструментов или в меню.

В двух строках инструментальных панелей на самом деле имеются 8 р ных панелей.

1. Главная панель.
2. Контроль симуляции.
3. Элементы оформления.
4. Арифметические блоки.
5. Логические блоки.
6. Приборы и пробники.
7. Источники сигналов.
8. Панель пользователя.

Инструментальные панели «Главная» и «Контроль моделирования» по ляются при запуске VisSim. Другие панели содержат кнопки блоков каж из соответствующих категорий меню Blocks. Инструментальная панель по зователя позволяет создать кнопки с нужными пользователю командами блоками.

2.6.4. Дерево структуры модели

Используемое окно VisSim разделено на две части. Левая область о отображает *дерево блок-схемы* или модели, т. е. является иерархическим пр ставлением последней. Наверху иерархии — имя файла рабочей блок-схе Дерево иерархии можно разворачивать/сворачивать, щелкая мышью по к ратикам ответвлений со знаком «+». Если ветвь открыта, то в квадратике является знак «–», что указывает на возможность закрытия ветви. Щелка именам ветвей, можно переходить внутрь одноименных составных блоков.

Границу между окном дерева структуры и окном модели можно пер щать мышью, указав линию границы курсором мыши и удерживая нажат левую клавишу. Таким образом, можно при необходимости убрать окно д ва структуры модели и предельно расширить окно модели.

2.6.5. Окно модели, полосы прокрутки и строка статуса

Окно модели — самая большая часть интерфейса системы VisSim. Пон лу оно пустое, и на нем присутствуют только линейки прокрутки. Они не ходимы для просмотра не умещающихся в окне программы больших мод (блок-схем). Правила работы с линейками прокрутки общеизвестны.

Если в окне модели введены блоки или модель загружена из файла окно будет заполнено блок-схемой всеми относящимися к ней элемент

блоками, соединениями, комментариями и т. д. При этом имеющуюся модель можно запускать на симуляцию (моделирование) и выполнять операции по ее модификации (редактированию). Эти операции, как и подготовку моделей, мы рассмотрим чуть позже.

Строка статуса системы VisSim содержит информацию о текущей блок-схеме. Отображаются следующие данные:

- количество блоков;
- временной диапазон моделирования;
- примененный метод интегрирования;
- размер шага интегрирования;
- имя неявного решателя.

В процессе моделирования в строке статуса отображается текущее значение времени. При перемещении мыши по элементам меню, кнопкам, блокам модели VisSim отображает краткое их описание в строке состояния.

2.6.6. Изменение вида интерфейса и позиция View меню

Позиция View (Вид) меню (рис. 2.18) позволяет изменять вид интерфейса. Управлять набором панелей инструментов можно также из контекстного меню правой клавиши мыши при установке курсора мыши на любую панель инструментов.

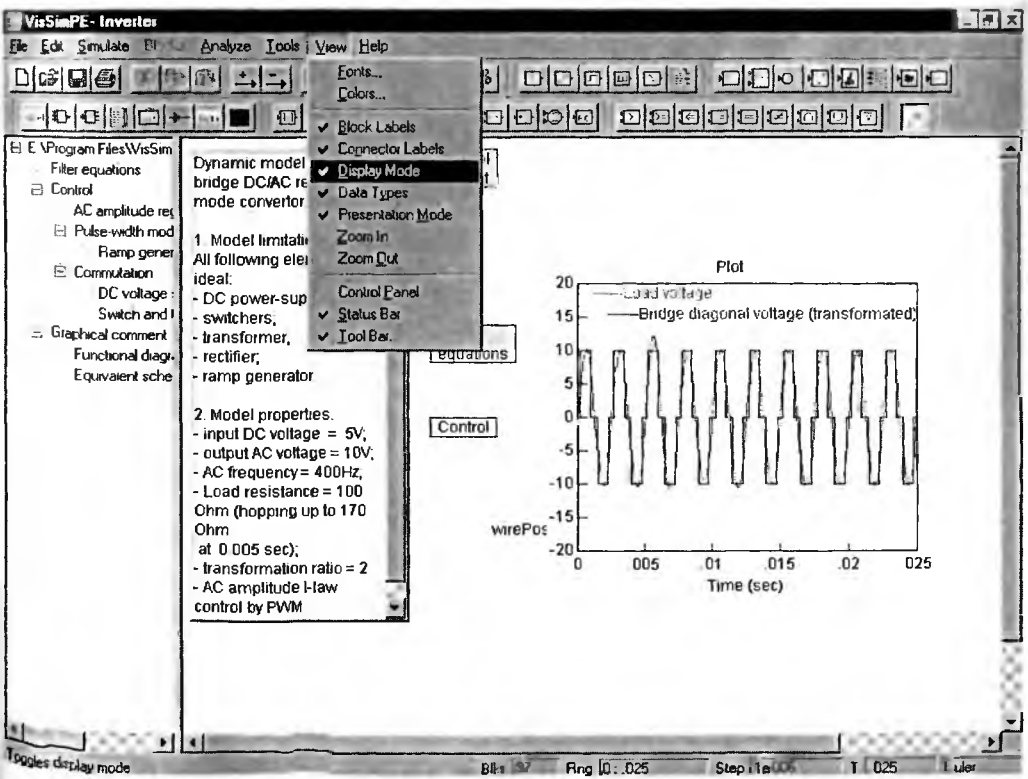


Рис. 2.18. Окно VisSim в режиме Display mode и с открытой позицией View меню

Первая команда Fonts... выводит стандартное окно смены набора шри тов, используемых для обозначений элементов модели. Это окно представл но на рис. 2.19 и позволяет выбрать тип, стиль и цвет заданного наб шрифтов. Особо надо отметить возможность выбора шрифтов по призна национальной принадлежности, например «Кириллица» для русского язы Это имеет особое значение при использовании русифицированных версий с стемы VisSim, которые иногда встречаются. Следует, однако, помнить, что гальные (оригинальные) версии VisSim — англоязычные. Тем не менее комментариев и обозначения блоков и соединений могут использоваться си волы кириллицы.

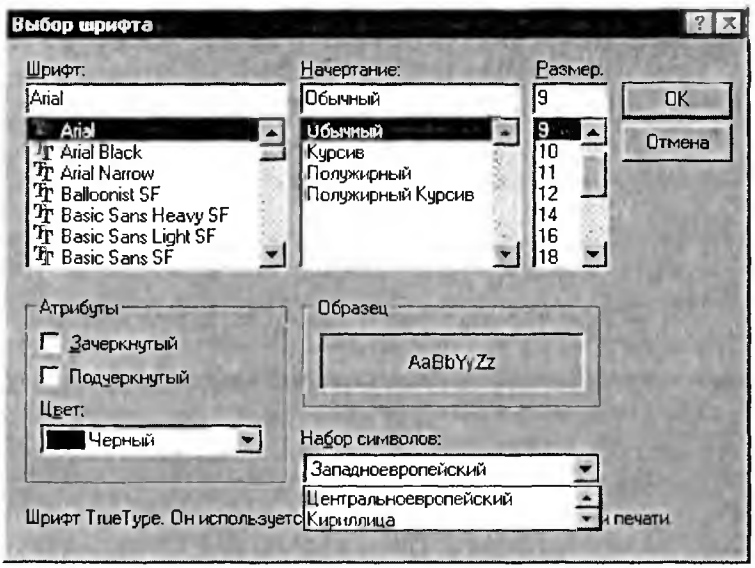


Рис. 2.19. Окно смены набора шрифтов

Команда Color... выводит окно смены цветов в цветовом оформлении дели. Это окно показано на рис. 2.20. Следует отметить, что выбор цвето VisSim выполнен очень тщательно. Как правило, изменение цветового офо ления лишь ухудшает его первоначальный вид. Оно рационально разве для пользователей, страдающих отсутствием нормального зрения.

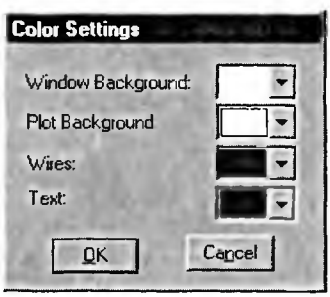


Рис. 2.20. Окно выбора цветов

Значительная часть команд меню View относится к включению или выключению отображения тех или иных панелей окна VisSim. Установив мышью знак птички против той или иной позиции подменю, можно установить соответствующую панель инструментов или блоков. Отсутствие птички означает, что соответствующая панель будет удалена. Это же правило относится и к командам Block labels (Показ меток блоков) и Connector labels (показ меток соединений). Показ меток блоков и соединений повышает информативность модели (блок-схемы), но при больших моделях может загромождать их.

Здесь же отметим, что изменить масштаб представления модели можно, используя команды Zoom In (Увеличение изображения в окне) и Zoom Out (Уменьшение изображения в окне). На размеры деталей интерфейса эти команды не влияют.

Особо надо отметить опцию Display mode. Если она введена, то из главной блок-схемы удаляются соединения и отображаются только блоки. Это хорошо видно из рис. 2.18 (сравните его с рис. 2.2, где режим Display mode отключен).

Нетрудно заметить, что в позиции View есть опции вывода/скрытия не всех панелей инструментов. Однако команда Tool bar... выводит показанное на рис. 2.21 окно с опциями всех панелей инструментов.

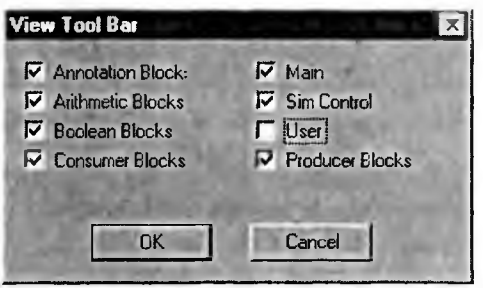


Рис. 2.21. Окно с опциями вывода/скрытия всех панелей инструментов

По умолчанию помечены все опции вывода панелей, кроме панели пользователя, создание которой мы отметим чуть ниже.

2.6.7. Перемещение инструментальных панелей

Любую панель инструментов можно выделить мышью и перенести в любое место окна. Для выделения мышью надо установить курсор мыши на панель за пределами кнопок. Это фиксируется черной рамкой вокруг панели. Далее, нажав левую кнопку мыши и удерживая ее, надо переместить ее в новое положение и зафиксировать в нем, отжав левую клавишу мыши. Именно так на рис. 2.22 показаны все панели, перенесенные со своего места в окно модели.

Нетрудно заметить, что при таком переносе панели инструментов приобретают титульную строку с названием соответствующей панели. На рис. 2.21 показана также панель Simulation Control (Контроль моделирования). Эта панель активизируется установкой птички у опции Control Panel в позиции View меню.

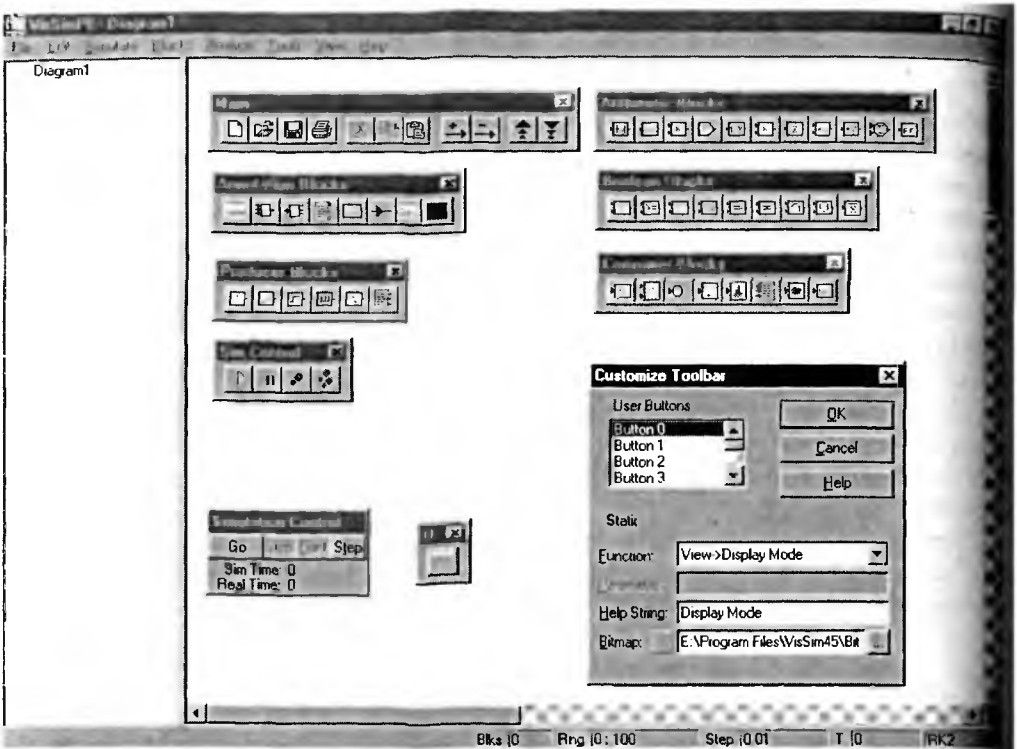


Рис. 2.22. Панели инструментов, перенесенные в окно модели

Двойным щелчком левой клавиши мыши при ее курсоре, установленно в любом свободном от кнопок месте панели (например, на титульной строк можно вернуть панель на место ее прописки.

2.6.8. Создание панели User Panel инструментов пользователя

Вы можете создать и настроить собственную панель инструментов U Panel. Для этого вначале в окне рис. 2.21 надо установить птичку у опц User. Далее надо в позиции Edit меню исполнить команду Edit Toolbar..., к торая выводит окно создания панели инструментов пользователя. Это ок показано на рис. 2.22 в правом нижнем углу.

В этом окне можно задать ряд кнопок, список которых обозначен к User Button (Кнопка пользователя). Далее надо в разделе Function выбрать обширного списка назначение кнопки, в строке Help String ввести строку по мощи данной кнопки, а в разделе Bitmap загрузить файл с графическим изо ражением создаваемой кнопки. VisSim поставляется с полным набором изо ражений всех элементов интерфейса и кнопок (папка Bitmap), но вы може создать свое изображение в формате BMP и использовать его для представл ния вашей кнопки. Некоторые кнопки при создании могут потребовать введ ния параметров в строке Parameter.

Таким образом, можно создать панель инструментов со своими кнопка (до 30). Работа с ней ничем не отличается от работы с встроенными панеля инструментов.

2.6.9. Режимы представления модели

Входы и выходы блоков в обычном режиме отображаются треугольниками, положение которых определяет направление распространения сигналов в блок-схеме. При активации режима презентации (опция Presentation Mode в позиции View меню) треугольники на входах блоков становятся меньше по размеру, а на выходах скрываются. Это делает блок-схему менее загроможденной и более презентабельной, но теряется ассоциативность при соединении блоков проводниками.

VisSim может выполнять функции *человеко-машинного интерфейса* (HMI-функции), например в случае применения в качестве программы, управляющей производственным процессом. Для этого VisSim может работать в специальном режиме Display Mode, который не позволяет конечному пользователю (обслуживающему персоналу) редактировать блок-схему, обрабатывающую сигналы от производственного процесса, т. е. перемещать блоки, изменять их соединения и параметры. Но у операторов остается возможность влиять на производственный процесс или на процесс исполнения блок-схемы посредствам управляющих элементов — регулятор, кнопка, константа.

Необходимо помнить, что активация режима Display Mode оказывает действие только на текущий уровень блок-схемы. Переход на другой уровень блок-схемы (для которого не активирован режим) позволяет вносить любые изменения. В режиме Display Mode видима только та часть блоков, которые могут быть использованы для оформления Display Mode. Провода и (по указанию) составные блоки скрываются. Режим также используется для анимационной визуализации результатов симуляции моделей.

При отладке проекта полезно контролировать тип данных обрабатываемых сигналов, поскольку существует класс ошибок, связанный с неверным выбором типов данных. VisSim обрабатывает четыре типа данных, каждому из которых соответствует определенная расцветка входов и выходов блоков:

1) число двойной точности с плавающей точкой — double (красный);
2) целое число со знаком — int (зеленый);
3) целое число без знака — unsigned int (синий);
4) шинный проводник (малиновый).

Если опция Data Types не активирована, то треугольные элементы у входов и выходов блоков не раскрашиваются. Отображение меток выводов у составных блоков задается активизацией опции Connector Labels.

2.6.10. Настройки среды VisSim

Оформление среды VisSim можно легко менять с помощью окна настроек, выводимого командой Preferences в позиции Edit меню (рис. 2.23).

На вкладке Preferences (Настройки) этого окна можно установить или убрать следующие опции:

- Show Horizontal Scroll Bar — показать горизонтальную линейку прокрутки;
- Show Vertical Scroll Bar — показать вертикальную линейку прокрутки;
- High Precision Display — установить высокое разрешение дисплея;
- Snap to Grid — освободиться от «сетки»;

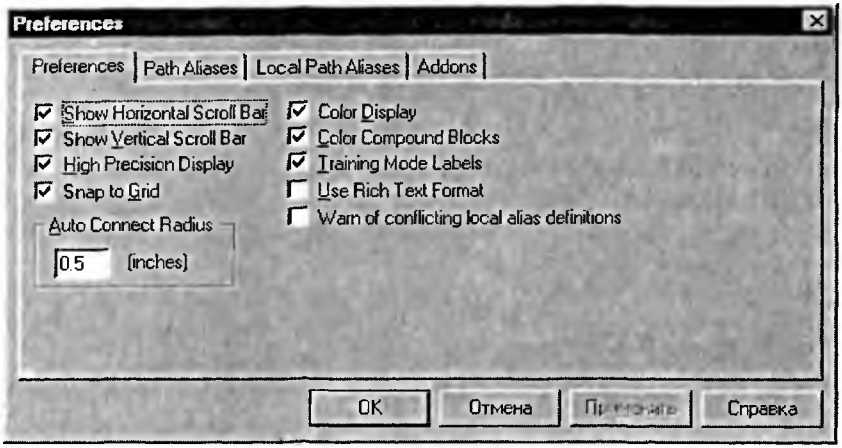


Рис. 2.23. Окно настроек среды VisSim

- Color Display — установить цветное изображение на дисплее;
- Color Compound Block — уставить цветное выделение блоков модели;
- Training Mode Labels — установить показ меток;
- Use Rich Text Format — использовать в комментариях Rich-формат текстов, принятый в текстовых процессорах Word;
- Warn of conflict local alias definition — оповещать о локальных конфликтах при иных определениях.

Отключение цветных режимов рационально, если используется печать монохромным принтером. В этом случае можно наблюдать модель и ее блоки в том виде, в котором они будут напечатаны.

VisSim имеет специальный обучающий режим Training Mode Labels, при включении которого ниже каждого блока отображаются следующие данные: его название, значения параметров или имя файла, с которым работает блок. Это делает диаграммы более понятными. Однако в случае больших диаграмм они оказываются перегруженными этими данными, и этот режим приходится отключать.

На вкладке Preferences есть также установка Auto Connect Radius радиуса обнаружения входов и выходов блоков в процессе их соединения мышью. В пределах этого радиуса происходит смена изображения курсора мыши с обычной стрелки на вертикальную стрелку или перекрестие. После такой смены можно нажимать левую клавишу мыши и начинать соединение между блоками.

2.7. Подготовка модели (диаграммы)

2.7.1. Создание новой модели

Создание новой модели (диаграммы или блок-схемы) начинается с использования команды New (Новая) в позиции File (Файл) меню или с активизации кнопки New в панели инструментов. Если вы работали над другой моделью и

не сохранили ее изменения, VisSim запросит вас о сохранении до создания новой диаграммы. Временно VisSim назовет диаграмму (файл) именем Diagram1.vsm. При первом сохранении VisSim предложит изменить название.

VisSim обеспечивает *визуально-ориентированный подход* к подготовке моделей. Она очень напоминает сборку электрической схемы из блоков конструктора. Нужные блоки модели выделяются на панели инструментов мышью путем указания на нужную кнопку курсором мыши и фиксации выбора коротким щелчком левой клавиши мыши. Можно выбрать также нужный блок и из меню Blocks.

После этого курсор мыши переносится в окно модели и приобретает вид блока, символически изображаемого прямоугольником из пунктирных черных линий. Он переносится мышью в нужное место экрана и фиксируется еще одним коротким щелчком левой клавиши мыши. После этого в окне подготовки модели блок принимает свой обычный вид — разный для каждого блока. Блок можно (при нажатой левой клавише мыши) перетаскивать с одного места окна в другое и точно позиционировать. Точность позиционирования обеспечивается привязкой к невидимой сетке.

Введя нужные блоки, их можно соединить опять-таки с помощью мыши. Возможны соединения без построения всей линии — достаточно пометить меткой начало соединительной линии и ее конец. Это упрощает построение сложных графических диаграмм. Процесс подготовки моделей также визуально-ориентированный, он довольно простой и наглядный. Позже мы рассмотрим его более подробно.

2.7.2. Установка параметров страницы модели

В начале работы с новой блок-схемой рекомендуется установить параметры страницы, исполнив команду Page Setup (Установка страницы) в позиции File меню. Все ваши настройки будут показаны на макете страницы в верхнем правом углу диалогового окна установки страницы (рис. 2.24).

Опции Orientation (Ориентация) позволяют выбрать ориентацию при выводе модели на печать. Они позволяют распечатать модель как лист книги (Portrait) или альбома (Landscape). В разделе установки полей можно задать размеры полей сверху, снизу, слева и справа (в дюймах). VisSim не отображает допустимую область для ввода блок-схемы, если не установлен режим предварительного просмотра печати. В этом режиме поля отображаются пунктирными линиями цвета морской волны.

Можно с помощью соответствующих текстовых переменных задать печать верхнего Header и нижнего Footer колонтитулов. Для записи в колонтитулы нужных данных могут использоваться следующие строковые переменные:

- Имя файла $f
- Путь к файлу $F
- Путь к блоку $H
- Дата $D
- Метод интегрирования $I
- Оптимизация $O
- Номер страницы $p

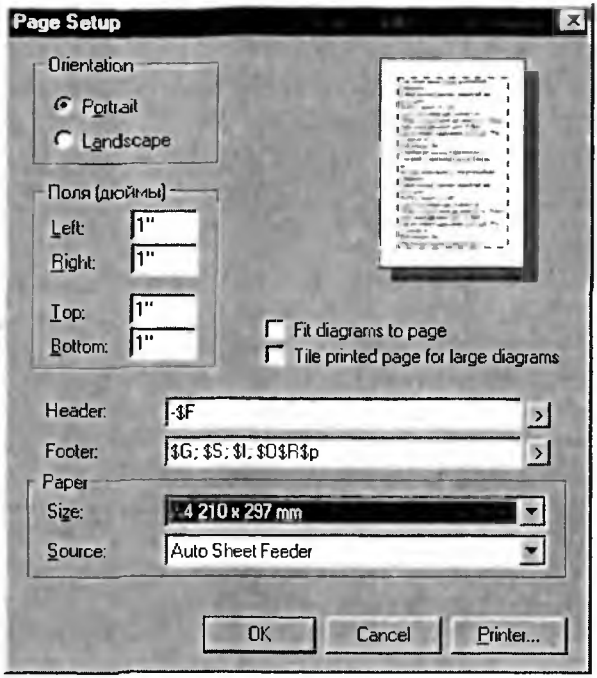


Рис. 2.24. Окно установок страницы

- Диапазон симуляции $G
- Размер шага $S
- Равнение по левому краю $L
- Центрирование $C
- Равнение по правому краю $R

В строках ввода колонтитулов можно вводить и обычный текст. Кроме того, можно задать формат (размер) бумаги Paper size и тип источника Paper Source.

Когда активизирована опция Fit diagram to page (Подравнять диаграмму на странице), VisSim печатает каждый уровень блок-схемы на отдельной странице. Если необходимо, VisSim уменьшает размер текста в блоках, так, чтобы уровень был размещен на одной странице в пределах указанных полей. VisSim печатает каждый уровень индивидуально с минимально возможным уменьшением размеров шрифтов, поэтому масштабы для уровней в многоуровневой блок-схеме могут оказаться разными. VisSim при включенной опции Fit diagram to page может оказаться не способным распечатать чрезмерно большие блок-схемы. В этих случаях VisSim даст вам возможность прервать операцию печати. Если вы проигнорируете предупреждение, VisSim напечатает ту часть блок-схемы, которая уместится на заданной бумаге с максимальным уменьшением

Опция Tile printed page for large diagrams (Печать для больших блок-схем) позволяет печатать каждый уровень без масштабирования на том количестве страниц, которое потребуется. Установки полей сохраняются в любом случае — умещается текущий уровень на странице или нет.

2.7.3. Открытие ранее созданной модели

VisSim имеет десятки уже подготовленных демонстрационных моделей (диаграмм). Кроме того, активно работающий с программой пользователь создает множество своих моделей. Их промежуточные или окончательные варианты записываются на те или иные устройства долговременной памяти, например на гибкие или жесткие диски. Вы можете быстро открывать любую из последних 12 блок-схем, над которыми вы работали ранее. Их имена отображаются в нижней части меню File.

Для *открытия* произвольной блок-схемы выберите команду Open (Открыть) в позиции File. Если до этого была открыта блок-схема, содержащая несохраненное редактирование, VisSim предварительно запросит вас о сохранении изменений. Если назначить название блок-схеме, используя команду Diagram Information... в позиции File меню, то оно будет появляться в диалоговом окне открытия файла в момент выбора блок-схемы.

Окно открытия файла ранее созданной модели уже описывалось (см. рис. 2.1). Это стандартное окно, применяемое во всех Windows-приложениях. Поэтому подробное его описание опущено.

2.7.4. Сохранение блок-схемы и ее передача по электронной почте

Текущая модель хранится в оперативном запоминающем устройстве. После ее редактирования и исполнения обычно требуется *сохранение модели* со всеми ее подмоделями и настройками в долговременном запоминающем устройстве компьютера — обычно гибком или жестком диске. Для этого используются самые обычные команды: Save для сохранения модели с текущем именем и Save As... для сохранения с новым именем. Команда Save As Metafile... используется для сохранения в формате мета-файла.

Команда Send позволяет передать модель по электронной почте в виде прикрепленного файла. Это возможно, разумеется, если на ПК установлен почтовый клиент, есть модем и подключение к Интернету. Это средство основывается на применении прикладого программного интерфейса передачи сообщений (Messaging Application Programming Interface — MAPI).

2.7.5. Предварительный просмотр модели и ее печать

Перед печатью модели принтером полезно просмотреть ее с помощью команды Print Preview... (Предварительный просмотр). Окно предварительного просмотра модели показано на рис. 2.25.

Для изменения масштаба области просмотра служат кнопки ZoomIn и ZoomOut. Кнопка Print служит для распечатки видимого изображения, а кнопка Close закрывает окно. Другие кнопки служат для переключения страниц в случае больших моделей.

В режиме предварительного просмотра страницы, используя возможности масштабирования, вы можете детально просмотреть макет. Пользуйтесь либо мышкой, либо кнопками Zoom In, Zoom Out.

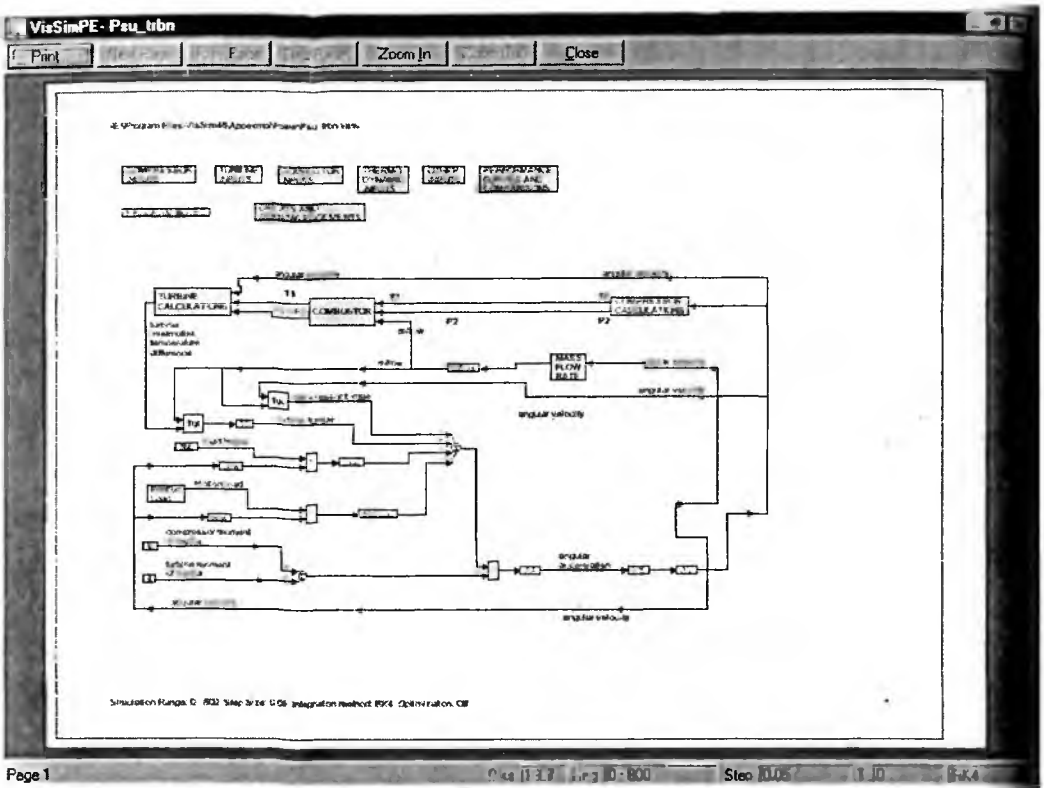


Рис. 2.25. Предварительный просмотр модели

Команда Print в позиции File меню выводит нестандартное окно печат (рис. 2.26). Окно содержит информацию о применяемом принтере и в разд ле Print Range позволяет задать печать всех страниц большой модели, страни цы текущего уровня и страницы текущего и более низкого уровня. Списо Copies (Копии) позволяет установить число копий для каждой из печатаемы страниц.

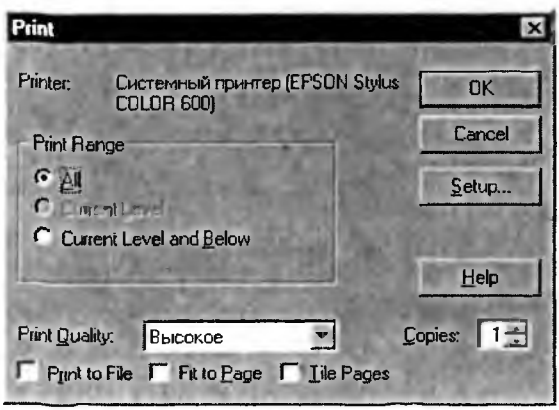


Рис. 2.26. Окно печати

Список Print Quality позволяет выбрать разрешение принтера. Есть также три дополнительные опции:

1) Print to File — печать в файл, который можно распечатать позже;

2) Fit to Page — печать с подгонкой страницы;

3) Tile Page — печать с последующей склейкой страниц в плакат.

Кнопка Setup..., а также команда Printer Setup... в позиции File меню выводит окно установок принтера, показанное на рис. 2.27. Это стандартное окно имеет список принтеров, что позволяет в случае необходимости использовать тот или иной доступный на вашем ПК принтер. Кнопка Свойства выводит окно свойств выбранного принтера. Вид этого окна зависит от применяемого драйвера принтера.

Рис. 2.27. Окно установок принтера

В окне рис. 2.27 можно выбрать размер бумаги и способ ее подачи. Возможна также установка книжной и альбомной ориентации модели на печатаемой странице.

2.7.6. Задание и получение информации о модели

Для задания и получения информации о модели надо исполнить команду Diagram Information... (Информация о диаграмме) в позиции File меню. Она выводит окно с информацией о текущей модели, с которой происходила работа в последнее время. Это окно показано на рис. 2.28.

В нижней части окна Statistics имеется «статистическая» (точнее общая) информация о модели. Так именуется имя файла, номер его версии, размер, дата последней модификации и общее количество блоков в модели и связанных с ней подмоделях. В верхней части имеются следующие поля:

- Title — ввод названия модели;
- Author — ввод данных об авторе;
- Comment — ввод текстового комментария.

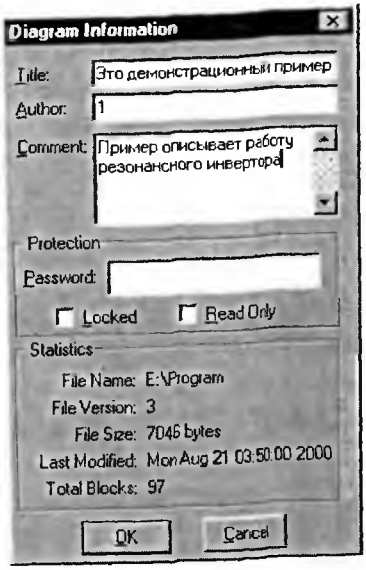


Рис. 2.28. Окно с информацией о текущей модели

В среднем разделе окна Protections (Защита) можно ввести пароль для з щиты данной модели от посягательств на нее других пользователей програ мой VisSim. Можно также задать статус файла модели как закрытого Lock или используемого только для чтения Read Only.

Для сохранения введенной информации о модели надо нажать кнопку О окна. Вся введенная информация будет отражена в последующем при загруз данной модели.

2.7.7. Выход из VisSim

Для *выхода* из VisSim используется команда Exit в позиции File мен При ее использовании VisSim проверит, была ли сохранена измененная м дель, и если была, то предложит сохранить ее перед выходом. Выход обеспе чивается также активизацией кнопки закрытия окна VisSim или применени комбинации клавиш ALT+F4.

В дальнейшем мы будем опускать описание комбинаций клавиш для т или иных команд, поскольку оно всегда указывается после наименовани команды в меню. Запомнить все комбинации трудно, да и вряд ли нужно, по скольку большинство команд продублировано кнопками панелей инструмен тов. Однако помнить некоторые наиболее часто встречаемые комбинации кла виш полезно, поскольку заметно ускоряет профессиональную работу с VisSim

2.8. Операции правки в позиции Edit меню

2.8.1. Общий обзор операций позиции Edit меню

Основные операции *правки* (редактирования) модели сосредоточены в по зиции Edit (Правка) меню, которая в раскрытом виде представлена на рис. 2.29.

Подменю этой позиции содержит ряд общепринятых команд редактирования и работы с буфером промежуточного хранения операционной системы Windows:

- Undo — отмена последней операции редактирования;
- Cut — удаление выделенного блока и размещение его в буфере;
- Copy — копирование выделенного блока в буфер;
- Paste — извлечение копии блока из буфера для размещения ее в окне модели;
- Paste Link — извлечение копии блока из буфера с организацией связи (обычно команда не задействована и отображается серым цветом).

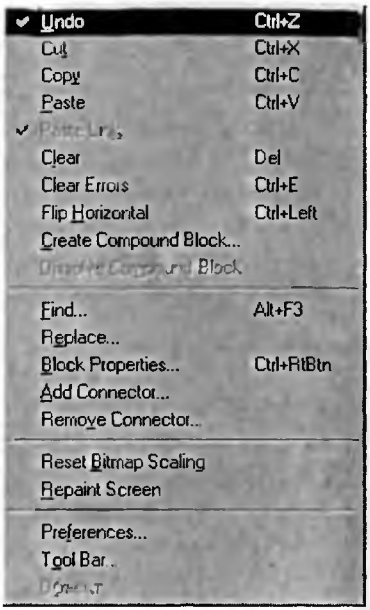


Рис. 2.29. Подменю позиции Edit меню

Ряд других операций характерен именно для системы VisSim:

- Clear — стирание выделенного блока;
- Clear Errors — сброс всех ошибок;
- Flip Horizontal — поворот блоков по горизонтали (выходы слева, входы справа);
- Create Compound Block... — вывод окна создания субблока;
- Dissolve Compound Block — отключение вычислений субблока.

Ряд команд связан с операциями редактирования блоков и соединений:

- Find ... — нахождение блока;
- Replace... — замена одного блока другим;
- Block Properties... — вывод окна свойств блока;
- Add Connector — добавление в блок одного или нескольких входов для соединений;
- Remote Connector — удаление входа с соединением;
- Reset Bitmap Scaling — удаления масштаба Bitmap объектов;
- Repaint Screen — перерисовка (обновление) экрана;
- Preferences... — открытие окна предпочтений;
- Tool Bar... — открытие окна подготовки панели инструментов пользователя;
- Object — обычно не используемая позиция подменю Edit.

Рассмотрим более подробно основные из этих команд и их применение для создания и редактирования моделей.

2.8.2. Выделение, перемещение и удаление блоков

Большинство из операций редактирования выполняется с выделенными блоками или их частями (например, входами и выходами). Для выделения отдельного блока или ряда блоков надо поместить курсор мыши в стороне, но близко от них и нажать левую клавишу мыши. Удерживая ее и перемещая

мышь, можно наблюдать появление прямоугольника из черных точечных линий. Надо добиться, чтобы прямоугольник охватил нужные блоки (рис. 2.30).

Затем надо отпустить левую клавишу мыши. Выделенные блоки окрасятся в черный цвет, что показано на рис. 2.31.

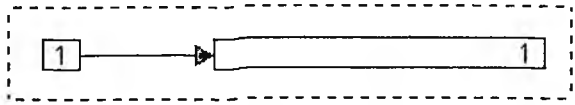


Рис. 2.30. Начало выделения блоков

Рис. 2.31. Конец выделения блоков

Теперь эти блоки можно скопировать в буфер командой Copy (горячие клавиши Ctrl+C) убрать с экрана и перенести в буфер командой Cut (Ctrl+X) или стереть командой Clear (Del). Команда Paste привязывает к курсору мыши блок, хранящийся в буфере. В окне модели блок превращается в прямоугольник из пунктирных черных линий. Его можно переместить в нужное место и, щелкнув левой клавишей мыши, зафиксировать в этом месте. При этом прямоугольник превратится в блок с соответствующим графическим представлением.

2.8.3. Некоторые операции редактирования и исполнения модели

Если вы изменили блок-схему, но решили отказаться от последнего действия, используйте команду Undo (Ctrl+Z). Команда Flip Horisontal используется для разворота блоков. Это особенно удобно, например, для просмотра виртуальным осциллографом сигналов с выхода блоков. В этом случае иногда полезно повернуть блок осциллографа (рис. 2.32).

Как уже отмечалось, команда Clear удаляет выделенные блоки и соединения. Но в отличие от команды Cut, которая размещает выделенные блоки в буфере, команда Clear содержимое буфера не меняет. Поэтому надо быть внимательным при применении этой команды.

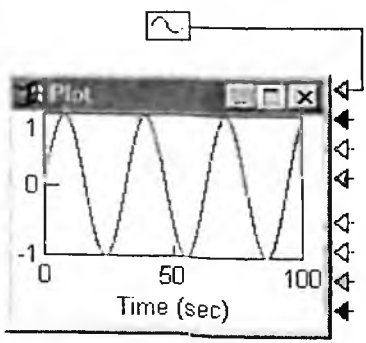


Рис. 2.32. Пример просмотра сигнала «перевернутым» блоком осциллографа

Иногда в ходе моделирования возникает ошибка. В этом случае блоки, в которых случается ошибка, «выставляют» флаги ошибок и закрашиваются красным цветом. Моделирование при этом прекращается. Команда Clear Errors сбрасывает флаги, и (после устранения ошибки) можно продолжить моделирование.

Иногда после редактирования на экране остаются остатки графического изображения некоторых объектов. Команда Repaint Screen обновляет изображение на экране и удаляет эти остатки.

2.8.4. Размещение и выделение блока

Размещение блоков в блок-схеме в основном зависит от предпочтений пользователя. Обычно оно должно быть таким, чтобы соединения между блоками были по возможности короткими и не пересекались. VisSim имеет ряд средств для создания «беспроводных» соединений, например это переменные, которые дают доступ к данным в любом месте блок-схемы. Набор блоков и их размещение, разумеется, прежде всего зависят от того, какую модель реализует пользователь. В пределах одного окна трудно разместить модель, имеющую больше двух-трех десятков блоков, поэтому большие модели создаются с применением подмоделей или субблоков.

При работе с блок-схемой часто требуется выделять тот или иной блок или совокупность блоков для различных манипуляций, например перемещения, копирования или удаления блока. Когда вы выделяете блок, VisSim подсвечивает его (фон блока становится темным). От текущих настроек среды зависит то, каким образом выделяются блоки. Если вы выделяете составной блок, то все составляющие блоки будут неявно выделены.

Для выделения блока выполните действия:

- расположите указатель мыши над блоком. Указатель сменится на крестик из стрелок;
- удерживая клавишу Shift, нажмите и тут же отпустите левую кнопку мыши;
- при повторении действий можно продолжить выделение по одному блоку.

Быстро выделить один или несколько блоков можно, используя область выделения, которая визуализируется пунктирным прямоугольником на поле блок-схемы при ее активизации. Та часть блок-схемы, которая будет находиться в области выделения, и те блоки, которых коснется ее граница, будет выделена. Для выделения блоков с помощью области выделения выполните действия:

- подведите указатель мыши к любому углу предполагаемой области выделения;
- нажмите и удерживайте левую клавишу мыши;
- перетащите указатель в противоположный угол области выделения, пока все выделяемые блоки не окажутся в постоянно обновляющейся области с пунктирной границей;
- отпустите левую кнопку мыши.

Иногда удобно инвертировать выделение на текущем уровне блок-схемы. Например, для снятия амплитудно-частотной характеристики разомкнутой системы можно выделить, удерживая клавишу Shift, генератор сигнала, главный сумматор и блок осциллографа, а затем инвертировать выделение. В результате выделенной окажется вся разомкнутая система из большого количества блоков. Для инверсии выделения на текущем уровне блок-схемы выполните действия:

- расположите указатель мыши над свободным пространством блок-схемы;
- удерживая клавишу SHIFT, щелкните левой клавишей мыши.

Отказ от выделения одного из блоков осуществляется следующим об разом:

- расположите указатель мыши над блоком. Вид указателя должен смени ться на крестик из стрелок;
- удерживая клавишу Shift, щелкните левой клавишей мыши.

2.8.5. Перемещение и копирование блоков

Перемещение и копирование блоков — это обычные операции при работ с блок-схемой в VisSim. Имеется несколько способов перемещения и копиро вания блоков. Например, можно перемещать блоки, перетаскивая их мышью или же вырезать их с копированием в буфер Windows, с тем чтобы затем вст вить в исходную или другую блок-схему VisSim или в другое Windows-прил жение.

Есть несколько правила перемещения и копирования блоков.

- Значения параметров блоков при их перемещении или копировани остаются неизменными и не сбрасываются к исходным значениям.
- Внутренние соединения в совокупности блоков при перемещении и копировании сохраняются.
- Внешние подключения к совокупности блоков сохраняются только пр перемещении. При копировании внешние подключения отсекаются.
- Если в размножаемой совокупности блоков имеется определение глоба льной переменной, то входной проводник к копии блока Variable (Пере менная) будет отключен. В составных блоках для обхода этого достато но локализовать переменную.

К простому способу перемещения или копирования блоков в пределах те кущего уровня блок-схемы относится метод *Drag-and-Drop* (Тяни и Отпускай Если вы перемещаете или копируете блоки в другой уровень или в файл дру гой блок-схемы, то необходимо использовать команды позиции Edit мен Copy, Cut и Paste. Эти команды используют буфер промежуточного хранени и обмена Windows. Скопировав (команда Copy) или перенеся (команда Cut) объект или группу объектов в буфер, можно вызвать их из буфера командо Paste в новом месте текущей диаграммы или в другой диаграмме, даже загру женной из файла.

Окно модели имеет невидимую сетку Grid. Для четкого позиционирова ния блоков в узлах сетки выполните действия:

- выберите команду позиции меню Edit команду Preferences и исполни те ее;
- в диалоговом окне этой команды выберите вкладку Preferences;
- активизируйте опцию Snap to Grid (Привязка к сетке);
- нажмите на кнопку OK или клавишу Enter.

Привязка к сетке включена по умолчанию, поскольку она облегчает точ ное позиционирование блоков в диаграмме. Если отказаться от такой привяз ки, то пользователю потребуются большие усилия по точному позициониро ванию блоков вручную с помощью мыши.

2.8.6. Поиск и замена блоков

В сложных моделях операции редактирования облегчаются применением типовых операций *поиска* и *замены* блоков. Для поиска блоков используется команда Find (Найти) в позиции Edit меню. Пример ее применения дан на рис. 2.33. В окне команды Find можно задать опции, уточняющие тип объектов поиска.

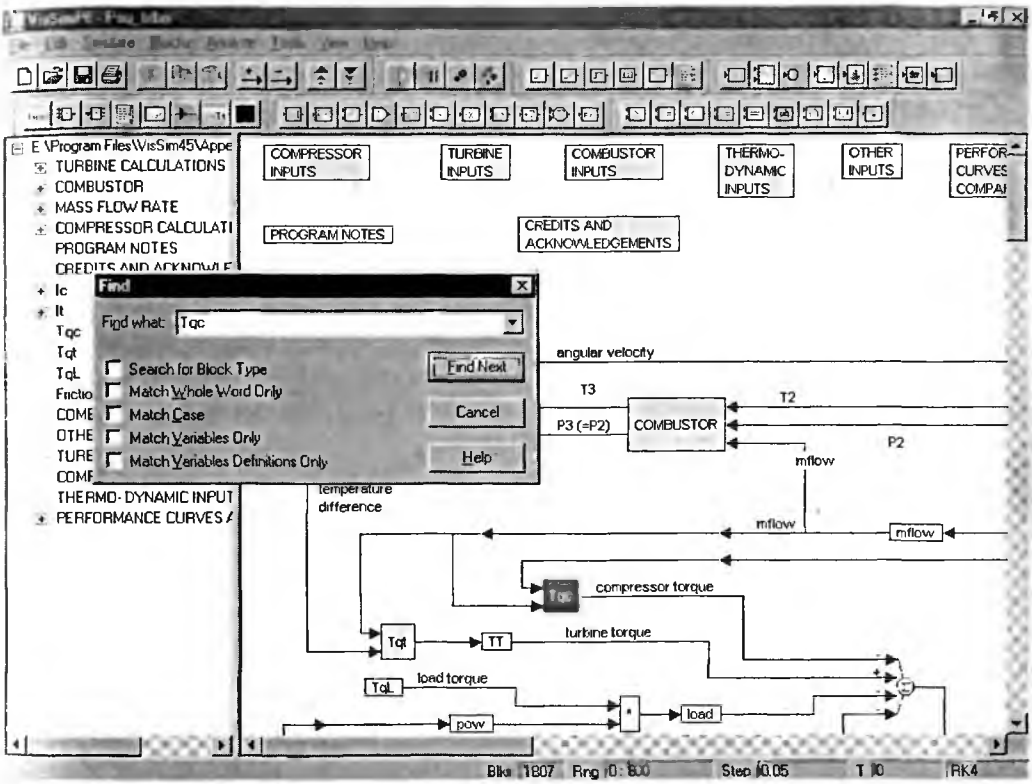


Рис. 2.33. Пример поиска блока Tqc с помощью команды и окна Find

Если нужно не только найти блок, но и заменить его другим (прямо скажем, не слишком частая операция), то можно использовать команду Replace. Стандартное окно этой команды показано на рис. 2.34. Наиболее типичное применение этой команды — замена слов в текстовых комментариях.

В случае замены блоков надо тщательно проверить функционирование модели. Бездумная замена чревата грубыми нарушениями работы модели.

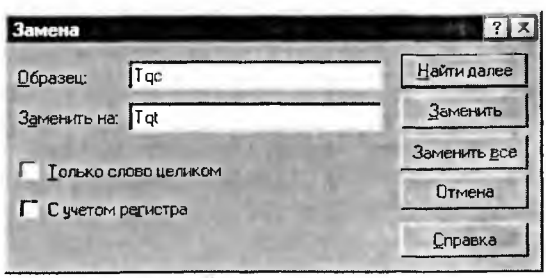


Рис. 2.34. Окно поиска и замены объектов

2.8.7. Вставка, настройки и соединение блоков

Для *вставки* блоков существует два основных способа:

1) блок выбирается с панели блоков щелчком левой клавиши мыши, переносится в нужное место окна модели и фиксируется повторным нажатием левой клавиши мыши;

2) блок выбирается из одной из групп библиотеки блоков в позиции Blocks меню, переносится в нужное место окна модели и фиксируется повторным нажатием левой клавиши мыши.

Блок обычно идентифицируется своим изображением, надписью внутри блока и меткой. Треугольники по сторонам блоков определяют входы и выходы для прохождения сигналов. Около контактов блоков могут присутствовать их имена для идентификации позиции сигналов в функции преобразования. Объем идентифицирующих данных зависит от режимов просмотра блок-схемы (модели).

Большинство блоков имеет устанавливаемые пользователем параметры, которые определяют их функции преобразования. Определить или изменить значения параметров можно с помощью одноименных диалоговых окон *свойств блоков* (Properties). Они появляются, если на блок навести курсор мыши и щелкнуть правой клавишей (или дважды левой). В дальнейшем (в главе 3) мы опишем диалоговые окна всех блоков системы VisSim.

Если вы поменяете параметры в момент выполнения симуляции, VisSim немедленно использует новые значения для отражения изменений. Начальные условия, которые устанавливаются в модели (обычно на блоках: 1/S, S&H, 1/Z, e-sTd и др.) в начале симуляции, также задаются через диалоговые окна свойств блоков. Окна свойств всех блоков описаны в главе 3. Каждое окно свойств имеет кнопку Help, открывающую раздел справки системы VisSim, относящийся к данному блоку и его применению.

Особо следует отметить правила ввода числовых данных. При вводе числовых данных VisSim отображает значения большие, чем 10^6, или меньшие, чем 10^{-6}, в показательной форме. VisSim использует латинскую букву e (или E) для отделения мантиссы от степени. Например, можно вводить число 6.000.000,0 следующими способами: 6e6 или 6E6.

Бывает удобно вводить числовые данные в параметры блоков с помощью простых математических выражений. Допустимо использование арифметических операторов: +, –, *, /; константы pi (π), функций и обычных правил приоритетов в расчетах. Например:

2 * (5 + 4) = 18

2 * 5 + 4 = 14

2 * pi * 50 = 314

В строках ввода числовых данных VisSim способен распознать выражения, написанные на языке СИ, результат которых, числовые данные. Вы можете использовать элементарные математические функции: acos, asin, atan2, cos, cosh, exp, abs, log, log10, pow, sin, sinh, sqrt, tan и tanh. Например, если вы введете pow(2,3) вместо коэффициента усиления в одноименном блоке (gain), то VisSim его вычислит (8) и примет к исполнению. VisSim также интерпрети-

рует символы pi как универсальную константу π = 3,1415926... Синтаксис языка СИ обычно хорошо изложен в любой книге по языку СИ.

Числовые расчеты модели всегда проводятся программой VisSim с точностью до 15 значащих десятичных цифр (речь о мантиссе). Однако при визуализации числовых данных можно ограничить мантиссу с 15 до 6 значащих десятичных цифр. Нужные установки можно найти в окне Preferences в позиции Edit меню.

Для большинства блоков в диалоговом окне свойств может быть определена текстовая метка Label. Когда активизирован соответствующий режим отображения блок-схемы (Block Label в позиции View меню), то метки появляются ниже соответствующих блоков вместо их параметров.

Соединение блоков проводниками указывает, в какой последовательности и какому блоку передать сигналы для обработки в течение времени моделирования по данной модели. Сигналы — это просто данные (значения координат модели). К входным сигналам (xn) относятся те, что поступают на входы блоков; выходные сигналы (yn) присутствуют на их выходах. В VisSim имеется два типа соединений (проводов):

1) простой проводник;

2) шинный проводник.

Простой проводник (FlexWire) — это тонкий провод, который позволяет передать только один сигнал (координату) между блоками. *Шинный проводник* на экране выглядит как более толстый, он содержит множество простых проводников и позволяет передавать совокупность сигналов. Шинные проводники используются при выполнении векторных или матричных операций или же в целях повышения наглядности проекта (незагромождения блок-схемы на верхнем уровне). Собрать одиночные проводники в шину и вывести их из нее можно при помощи парных блоков scalarToVec и vecToScalar соответственно.

Удобной альтернативой для передачи сигналов может являться возможность использования определяемых пользователем переменных. Этот механизм подобен использованию символа заземления в схемах электрических принципиальных, и его главное достоинство аналогично — повышение наглядности блок-схемы за счет сокращения загромождающих ее проводниковых связей.

Вы можете прикреплять проводники и шины к выводам блоков. Как только к блоку подключается соединение (провод или шина), VisSim принимает связь и обслуживает ее в дальнейшем при моделировании. Если вы перемещаете блок по рабочему полю, то проводники автоматически перерисовываются (и не отключаются).

Надо учитывать следующие правила соединений:

- соединить можно только пару — вход и выход;
- к любому входу можно подключить только один проводник (сигнал);
- к любому выходу можно подключить множество входов (проводников);
- VisSim автоматически располагает провода по кратчайшему пути.

Треугольная форма оформления выводов, указывающая направление прохождения сигналов, позволяет легко отличить входы от выходов. VisSim при заданном расположении блоков задает кратчайший путь соединений, но только с применением горизонтальных и вертикальных линий. При этом нередко

линии соединений пересекают блоки. Специальный блок writePositioner позволят проложить соединение любым другим способом (см. главу 3).

Для облегчения процесса соединения блоков определен радиус зоны с центром на выводах блоков. При доведении проводника до этой зоны автоматически происходит подключение. Если зоны входов и выходов перекрываются, то для их соединения достаточно просто указать мышкой любой из выводов. Радиус зоны соединения задается в окне Preferences.

Для удобства обзора модели используется цветовая раскраска проводников. Ее можно изменить в окне изменения цветов, выводимых командой Color... в позиции View меню. Для этого можно назначить класс соединений, соответствующих выводам блоков.

При активизации режима Display Mode VisSim скрывает все проводники. Этот режим предназначен для отображения контрольной или контрольно-измерительной панели без проводниковых связей или когда требуется визуализировать анимацию.

Для удаления проводника необходимо отделить его от входа блока, к которому он подключен. Для этого достаточно указать мышью подключенный вход и, удерживая нажатой левую клавишу мыши, отвести проводник от входа. Отпустив кнопку мыши, можно увидеть удаление проводника (соединения).

Все блоки, которые обрабатывают сигналы в VisSim, имеют выводы — входы и выходы. VisSim различает входы и выходы и использует их по-разному. Входы предназначены для передачи сигналов функциям преобразования блоков. На выходы поступают результаты выполнения блочных функций. Треугольная форма оформления выводов (контактов) блоков позволяет вам легко проследить направления распространения сигналов по блок-схеме.

Некоторые блоки имеют символы над выводами, которые указывают тип данных или каким образом блочные функции обрабатывают сигналы. Например, «~» над входом блока сумматора означает инверсию сигнала. Можно добавлять или удалять выводы у большинства блоков VisSim. Если на момент удаления вывода к нему подключен проводник, то он так же будет удален с блок-схемы. Не забывайте достраивать внутреннюю структуру составных блоков, к которым вы добавляете новые входы или выходы. Вы можете назначать имена выводам составных блоков.

Для добавления или удаления вывода блока выполните следующие действия.

- Воспользуйтесь нужной кнопкой в главной панели инструментов или выберите команду Add Connector (Добавить Соединение) или Remove Connector (Удалить Соединение) в позиции Edit меню. Указатель мыши приобретет вид, подобный изображениям на этих кнопках.
- Подведите указатель к выводам блока и зафиксируйте момент, когда указатель превратиться в горизонтальную черточку (при добавлении выводов) или выводы блоков будут выделяться пунктирным прямоугольником (при удалении выводов).
- Определитесь с местом добавления вывода или с удаляемым выводом и нажмите левую клавишу мыши.
- Повторите эти действия для других выводов блока.

- Переместите указатель мыши на пустое место блок-схемы и нажмите левую клавишу мыши для завершения операции.

Так называемые классы соединений в VisSim обеспечивают ассоциативное оформление расчетов координат в блок-схеме по именам и цвету. Определить и назначить класс для соединений можно посредством окна свойств соединений Connector Properties. Для его вызова следует дважды щелкнуть мышью по выводу блока. Вид окна показан на рис. 2.35.

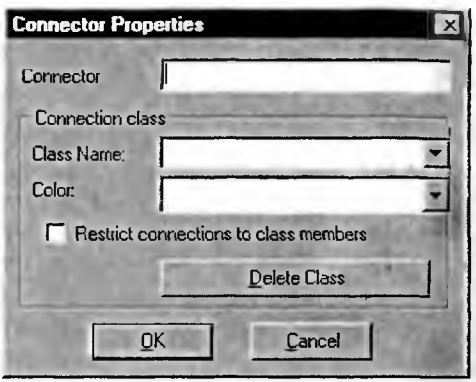


Рис. 2.35. Окно задания класса и свойств соединения

Класс соединения определяется его именем и цветом проводников. Например, вы можете назначать имя класса «Напряжения» тем цепям блок-схемы (выходам), которые связаны с расчетом напряжений. Если класс назначен определенному выходу, то куда бы он ни был подключен, проводники будут окрашены в цвет класса. Для задания класса в окне рис. 2.35 используется открывающийся список Class Name (поначалу он пуст), а для изменения цвета — список цветов Color. Разумеется, вы можете отказаться от этих дополнительных хлопот и положиться на классы и цвета соединений, принятых в системе VisSim по умолчанию.

2.9. Установка свойств моделирования

2.9.1. Команды позиции Simulate меню

После подготовки модели она может быть запущена, т. е. можно начать собственно моделирование. При этом VisSim автоматически составляет и решает систему алгебраических и дифференциальных уравнений для заданной модели и выводит результаты моделирования в виде показаний назначенных пользователем виртуальных измерительных приборов. Часто модель начинает работать сразу в соответствии с установками, принятыми по умолчанию.

Однако чаще необходима некоторая дополнительная настройка — как параметров блоков модели, так и ее самой. Пользователь должен четко понимать, что временные параметры модели обычно не относятся к параметрам реального времени. Время моделирования одной и той же модели может резко различаться у разных по быстродействию ПК.

Для управления процессом моделирования служит позиция Simulate меню. Она имеет следующие команды:

- Go — пуск моделирования;
- Stop — остановка моделирования;
- Single Step — пуск очередного шага при пошаговом моделировании;
- Continue — продолжение моделирования после остановки;

- Reset Sim — сброс моделирования;
- Simulation Properties ... — вывод окна свойств (параметров) моделирования;
- Optimization Properties ... — вывод окна свойств оптимизации (см. раздел 3.10);
- Snap States — моделирование с запоминанием состояния;
- Reset Srates — сброс состояния.

Действие этих команд вполне очевидно. Тем не менее некоторые и наиболее важные из них мы рассмотрим ниже более подробно.

2.9.2. Установка времен и режимов моделирования

На вкладке Range (рис. 2.36) окна свойств моделирования можно установить следующие временные параметры моделирования:

- Start — начальное время моделирования;
- Step Size — размер шага моделирования;
- End — конечное время моделирования.

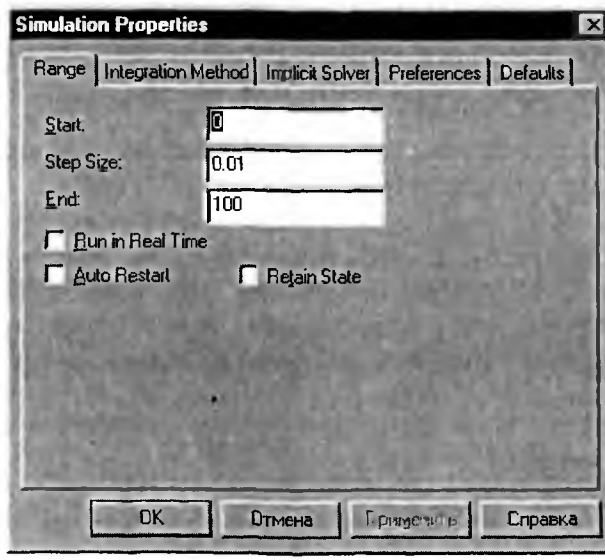


Рис. 2.36. Вкладка Range окна свойств моделирования

Размер шага Step Size играет огромное значение для процесса моделирования. Он указывает интервал, по прошествии которого алгоритм метода интегрирования обновляет значение интеграла входной функции, а измерительные блоки регистрируют результаты моделирования. При большом шаге моделирования оно может нередко пойти вразнос, порождая результаты в виде несуществующих колебаний тех или иных величин. Возможна даже аварийная остановка моделирования из-за переполнения разрядной сетки чисел, используемых при моделировании. При этом блоки, в которых это произошло, выставляют флаги ошибок и окрашиваются в красный цвет. Для сброса ошибок служит команда Clear Errors в позиции Edit меню.

Если решение расходится, то прежде всего надо попытаться уменьшить временной шаг интегрирования. Однако это неизбежно приводит к увеличению числа шагов моделирования и увеличению его общего времени. Поэтому в моделях, результаты моделирования которых представляются кривыми с разными (медленными и быстрыми) участками, полезно применение адаптивных методов интегрирования, допускающих автоматическое изменение шага интегрирования. К таким методам относятся адаптивный метод Рунге—Кутта 5-ого порядка и адаптивный метод Булирша—Стоера. Для них можно определить минимальный размер шага.

Иногда, например, при использовании блоков систем Mathcad и MATLAB, на которых возложены основные операции моделирования, желателен однократный запуск модели. Для этого достаточно установить начальное время моделирования Start = 0 и задать Step Size = End = 1. При этом будет выполнен только один шаг моделирования.

Настройку интеграторов VisSim мы рассмотрим чуть позже, а пока отметим, что на вкладке Range можно также задать ряд опций режимов моделирования:

- Run in Real Time — режим расчетов в реальном времени (или свободный с максимальной производительностью компьютера);
- Auto Restart — задание автоматического перезапуска моделирования (режим «непрерывного» моделирования);
- Retain State — сохранение данных предшествующего моделирования.

С активной опцией Run in Real Time VisSim выполняет моделирование в реальном масштабе времени, т. е. одна секунда моделирования будет равна часовой секунде. Основное назначение этого режима — работа с аппаратными средствами, управляемыми моделью VisSim. Для этого необходим пакет расширения Vissim/Real-Time и специализированная плата ввода/вывода для управления внешними устройствами. У нас такие платы крайне редки. Специальный драйвер Vissim/Real-Time позволяет конфигурировать порты аналоговых и цифровых каналов на платах разных производителей и имеет блоки чтения и записи для подключения к блок-схеме.

Режим автоматического рестарта Auto Restart обычно нужен для управления реальным объектом или для облучения нейронных сетей. Прервать цикл автоматически может поступление единичного сигнала на блок Error (Ошибка) или поступление сигнала большего 2 на блок Stop (Стоп). Можно, разумеется, и вручную остановить моделирование. Номер моделирования хранит предопределенная переменная $runCount (VisSim имеет ряд системных переменных, отличительным признаком которых является символ $ в начале имени).

Для сохранения текущих координат системы при перезапусках следует активизировать опцию Retain State. Однако есть блоки (например, в группе генераторов), чьи сигналы зависят от времени моделирования и сбрасываются при рестарте. Для корректных переходов в режиме Auto Restart надо удалить из модели блоки, генерирующие сигналы. Например, если модель содержит блок синусоида, следует заменить его интегратором и подать выходной сигнал на синусоидальный преобразователь.

2.9.3. Выбор метода и шага интегрирования

Для интеграторов (блоки с обозначением 1/S) возможен выбор различных алгоритмов *численного интегрирования*. Простые алгоритмы с неизменным шагом интегрирования обеспечивают высокую скорость интегрирования, но могут привести моделирование к разносу — числовой нестабильности. Если выбирается адаптивный алгоритм с переменным шагом, то надо дополнительно определить минимальный размер шага, допуск ошибки и количество возможных итераций, иначе моделирование может зациклиться.

VisSim имеет семь алгоритмов интегрирования различной точности для численного решения (методом интегрирования) дифференциальных уравнений:

- метод Эйлера;
- метод трапеций;
- метод Рунге—Кутта 2-го порядка;
- метод Рунге—Кутта 4-го порядка;
- адаптивный метод Рунге—Кутта 5-го порядка;
- адаптивный метод Булирша—Стоера;
- обратный метод Эйлера для жестких систем дифференциальных уравнений.

Установка нужного метода интегрирования выполняется в окне свойств моделирования на вкладке Integration Method. Вид этой вкладки показан на рис. 2.37. Для выбора метода достаточно мышью поставить жирную точку против имени метода.

Рассмотрим кратко особенности применяемых методов (более подробное рассмотрение есть в обширной литературе по численным методам).

Метод Эйлера — это простой одношаговый метод интегрирования. Оценка значения вычисленного интеграла выполняется один раз за шаг симуля-

Рис. 2.37. Вкладка Range окна свойств моделирования

ции. При оптимальном выборе шага метод обеспечивает самую высокую скорость моделирования. Но он критичен к выбору шага и нередко приводит к расхождению решения. Да и точность метода явна невысока.

Метод трапеций имеет на порядок меньшую погрешность. Оценка значений интеграла выполняется дважды за шаг моделирования. Этот метод пригоден для решения многих задач моделирования, если его результаты представимы гладкими и не слишком быстро изменяющимися кривыми.

Метод Рунге—Кутта 2-го порядка имеет второй порядок. Он вычисляет производную в середине шага для оценки значения интеграла в конечной точке шага. По точности вычислений он эквивалентен методу трапеций.

Метод Рунге—Кутта 4-го порядка имеет погрешность на два порядка меньше, чем у методов трапеций и РунгеКутта 2-ого порядка. Метод рассчитывает производную четыре раза на каждом шаге: в начальной точке, дважды в середине шага и в конце шага. Результаты используются для оценки значения интеграла. Это надежный и апробированный метод, пригодный для решения большинства задач моделирования, за исключением жестких.

Адаптивный метод Рунге—Кутта 5-го порядка имеет точность пятого порядка. Адаптивность метода заключается в том, что при ускорении изменений входных координат алгоритм автоматически уменьшает размер шага. Это уменьшает вероятность «разноса» решений. Разумеется, приспособление метода к особенностям решения дифференциальных уравнений в ходе моделирования даром не достигается — этот метод обычно замедляет моделирование.

Адаптивный метод Булирша—Стоера использует полиномиальную экстраполяцию для оценки значения интеграла в конечной точке шага на основе серии предыдущих значений. Алгоритм обладает малой погрешностью для гладких функций (в типовых режимах движения, когда координаты меняются с постоянной скоростью, ускорением или приращением ускорения). Это достаточно эффективный современный метод интегрирования.

Обратный метод Эйлера наиболее эффективен для моделирования жестких систем с большой разницей частот собственных колебаний или постоянных времени моделей. К таким системам относятся электронные цепи, модели химических и иных процессов. Другие методы требовали бы установки существенно меньшего размера шага для устойчивого (без разноса) моделирования.

Вкладка выбора метода интегрирования имеет также установку трех временных параметров для шагов интегрирования. Адаптивные алгоритмы (Рунге—Кутта 5-го порядка и Булирша—Стоера) могут менять величину шага симуляции. Размер шага непрерывно корректируется, чтобы попасть в допуск ошибки за ограниченное количество итераций. Для этих методов можно задать минимальный размер интегрирования Min. Значение его по умолчанию — 1e-006.

При выборе этих методов интегрирования можно также задать максимальную ошибку между результатами двух последовательных итераций. VisSim использует эту ошибку для определения размера шага. Чем больше допуск ошибки, тем больший размер шага возможен. Значение по умолчанию — 1e-005. Можно также задать максимальное количество попыток (итераций)

изменения величины шага симуляции в целях уменьшения ошибки обрыва. Значение по умолчанию равно 5.

2.9.4. Установки решателей импликативных уравнений

В ходе моделирования нередко приходится решать неявные и *импликативные* уравнения, например алгебраические. Установки параметров решателей таких уравнений представлены на вкладке Implicit Solver (рис. 2.38).

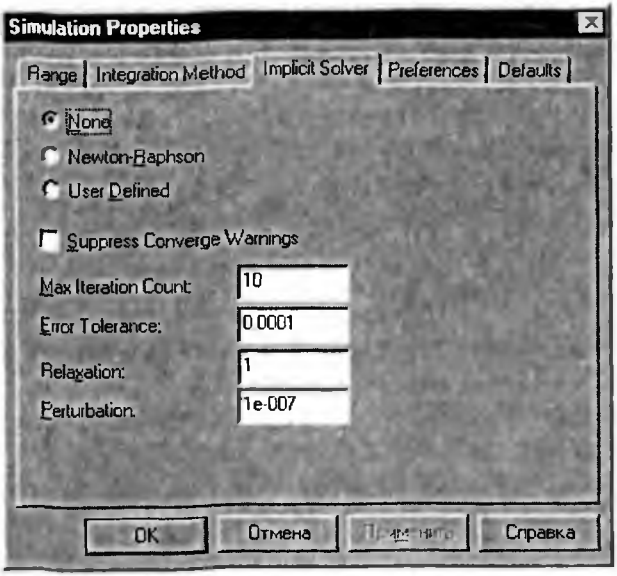


Рис. 2.38. Вкладка Implicit Solver окна свойств моделирования

На этой вкладке можно установить три основные опции:

1) None — отказ от применения решателя импликативных уравнений;

2) Newton-Raphson — применение решателя, реализующего *метод Ньютона—Рафсона*;

3) User — применение заданного пользователем решателя.

Выбор той или иной опции зависит от характера решаемой задачи.

2.9.5. Установка предпочтений моделирования

Вкладка Preferences (Предпочтения) окна свойств моделирования, показанная на рис. 2.39, позволяет активизировать режим фиксации состояния модели в файле, настроить механизм уведомлений пользователя, установить опции предпочтений, определить сценарий запуска программы и изменить базу псевдослучайных последовательностей у генераторов случайных чисел. Все эти возможности реализуются с помощью опций, рассмотренных ниже.

Опция Checkpoint State сохраняет состояния блок-схемы в момент приостановки моделирования. В частности, VisSim сохраняет текущие значения

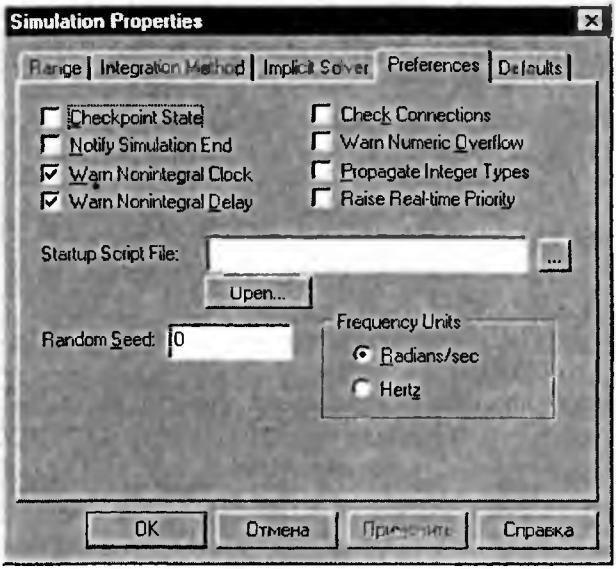


Рис. 2.39. Вкладка Preferences окна свойств моделирования

всех координат системы, треки визуализации приборов и текущее время симуляции. Если блок-схема закрывается, то при ее повторном открытии можно будет продолжить моделирование с момента приостановки. Этот режим особенно полезен для продолжительного моделирования. Так, он дает возможность остановить процесс, когда необходимо выключить компьютер, с сохранением состояния блок-схемы. Однако опция должна быть активизирована прежде, чем вы начинаете моделирование по заданной модели.

Опция Check Connectors активизирует вывод предупреждений перед выполнением моделирования в случае, если в блок-схеме имеются блоки с неподключенными входами. Такие блоки будут помечаться красным цветом. Появляющееся диалоговое окно позволяет аварийно прекратить моделирование, проигнорировать первый найденный неподключенный вход и продолжить проверку блок-схемы и проигнорировать все неподключенные входы и выполнить моделирование.

Опция Warn Nonintegral Delay задает вывод предупреждения для случаев, когда в блок-схеме для блока временной задержки получено значение, не кратное шагу симуляции. Желательно, чтобы опция была всегда активна, иначе ошибка моделирования может оказаться необнаруженной.

Опция Warn Nonintegral Clock задает вывод предупреждений в случаях, когда в блоках модели, например в блоках синхроимпульсов, установлен период синхронизации, не кратный шагу симуляции. Желательно, чтобы опция была также всегда активна.

Опция Warn Numeric Pverflow задает выход предупреждений в случае переполнения результатов численных вычислений.

VisSim использует правила языка СИ, чтобы преобразовать младшие целочисленные типы данных к старшим. Опция Propogate Integer Types активи-

зирует вывод предупреждений о таких событиях и позволяет держать их под контролем.

Опция Notify Simulation End вызывает появление информационного диалогового окна в конце моделирования с сообщением об этом событии.

Область вкладки Frequency Unit позволяет выбрать единицу измерения частоты (в рад/с и Гц) для восприятия данных в строках ввода параметров блоков, например в блоках синусоиды, и задания передаточных функций.

Опция Random задает начальное число для генераторов случайных чисел, что обеспечивает создание новой их последовательности. Начальное число может принимать значения 0 до 65535. Значение по умолчанию — 0. Опция Random активизируется в случае управления реальным объектом из VisSim. Вашему процессу присваивается более высокий приоритет для надежного осуществления процессов выборки и записи в каналы специализированных плат ввода/вывода, в реальном масштабе времени, без прерываний от других процессов, которые выполняются одновременно.

2.9.6. Настройки моделирования по умолчанию

Вы можете определить назначаемые по умолчанию настройки моделирования при запуске VisSim или при создании новой блок-схемы. К таким настройкам относятся: метод интегрирования, начальное и конечное время моделирования, шаг интегрирования и максимальное число точек для кривых графопостроителя (осциллографа). Они размещены на вкладке Default, представленной на рис. 2.40.

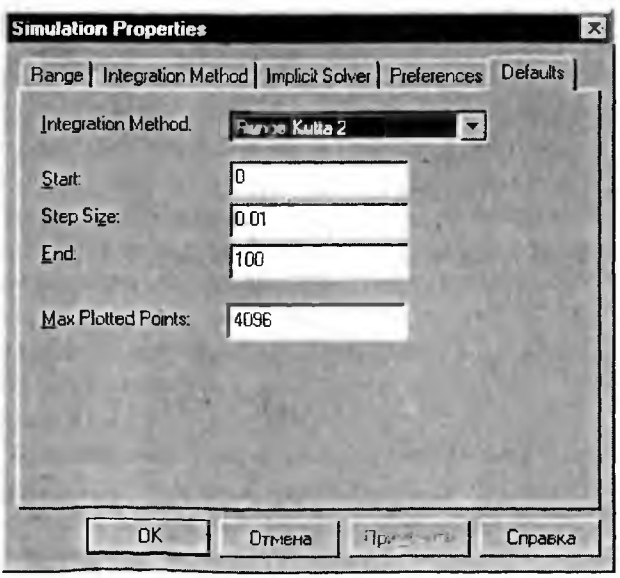


Рис. 2.40. Вкладка Default окна свойств моделирования

Выбор метода интегрирования осуществляется из списка. Для установки любого другого нужного параметра просто введите его значение. Зафиксируйте ввод нажатием клавиши OK.

2.9.7. Системные переменные VisSim

В системе VisSim определены системные переменные, имя которых начинается с символа $. Ниже представлен их список:

- $firstPass — задает начальный импульс для первого шага моделирования;
- $lastPass — задает конечный импульс для последнего шага моделирования;
- $randomSeed — задает установку генератора случайных чисел;
- $runCount — задает задержку итераций моделирования при реализации метода Монте-Карло и переключений параметров;
- +$timeStart — возвращает начальное время моделирования;
- $timeStep — возвращает шаг во времени моделирования;
- $timeStop — возвращает конечное время моделирования.

Значения системных переменных можно присваивать как значения блока констант const и проверять их с помощью блока display. Это позволяет использовать системные переменные для управления вычислительным процессом, а также в тех случаях, когда задаваемые ими параметры нужны для работы каких-либо блоков VisSim.

2.10. Работа со справкой VisSim и демонстрационными примерами

2.10.1. Меню справки Help

Для облегчения работы с системой VisSim она оснащена справочной системой (*справкой*) и обширным набором демонстрационных примеров. Операции со справкой сосредоточены в позиции меню Help, показанной на рис. 2.41.

Подменю справки содержит следующие команды:

Context ... — вывод окна справки по контексту;

Search... — вывод окна стандартной онлайновой справки;

Wiring... — справка по соединениям блоков;

Simulating... — вывод окна с данными о методике моделирования;

Using Help... — вывод окна справки по справке;

Web Updates ... — вывод окна обновления VisSim через Интернет;

About VisSim... — вывод окна с данными о программе VisSim.

Рис. 2.41. Подменю Help меню системы VisSim 4.5

Рассмотрим некоторые особенности работы со справкой системы VisSim.

2.10.2. Справка по контексту

Исполнив команду Context, можно вывести окно справки по контексту. Это окно представлено на рис. 2.42. Как обычно, справка реализована с применением гипертекстовых ссылок, в роли которых выступают названия разделов справ-

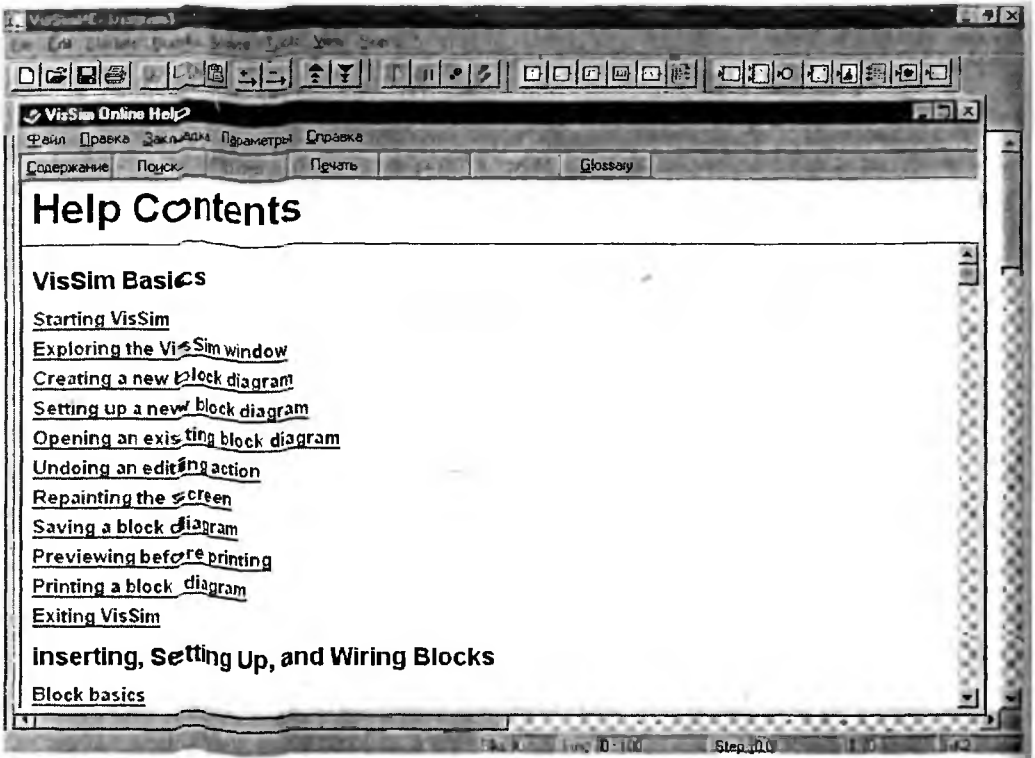


Рис. 2.42. Работа со справкой VisSim по контексту

ки. Естественно, что справка реализована на английском языке. Русскоязычные надписи для элементов интерфейса окна справки указывают на то, что используются окна справки русифицированной операционной систем Windows.

Для получения справки по нужному разделу надо активизировать мышью соответствующую гипертекстовую ссылку, т. е. поместить на нее курсор мыши и щелкнуть ее левой клавишей. Появится окно справки выбранного раздела. На рис. 2.43 для примера показано полностью открытое окно справки по библиотеке блоков VisSim. В нем видны названия разделов библиотеки.

Если материал справки не помещается в окне раздела (что характерно для окна, показанного на рис. 2.43), то следует воспользоваться линейкой прокрутки. Рисунок 2.44 показывает конец справки по разделу о библиотеке блоков. В конце этого раздела виден ряд дополнительных гипертекстовых ссылок, позволяющих расширить объем справки по выбранному разделу.

Если активизировать эти гипертекстовые ссылки, то появляются всплывающие окна на фоне окна раздела справки. Одно из таких окон показано на рис. 2.44 справа. Эта возможность повышает удобство работы со справкой, поскольку после закрытия всплывающего окна справки вы оказываетесь в последнем окне раздела.

Окна справки являются самостоятельными окнами. Их можно открывать, закрывать и сворачивать в бирку, размещаемую в панели задач Windows. Окна справки можно также перемещать мышью и растягивать в том или ином направлении.

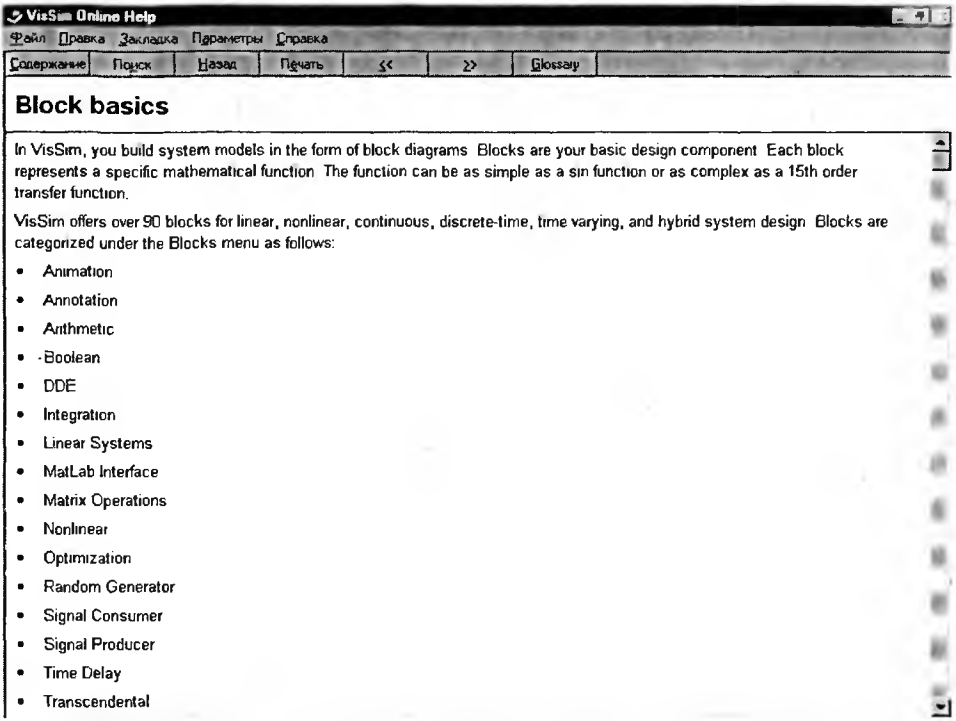


Рис. 2.43. Окно справки по библиотеке блоков

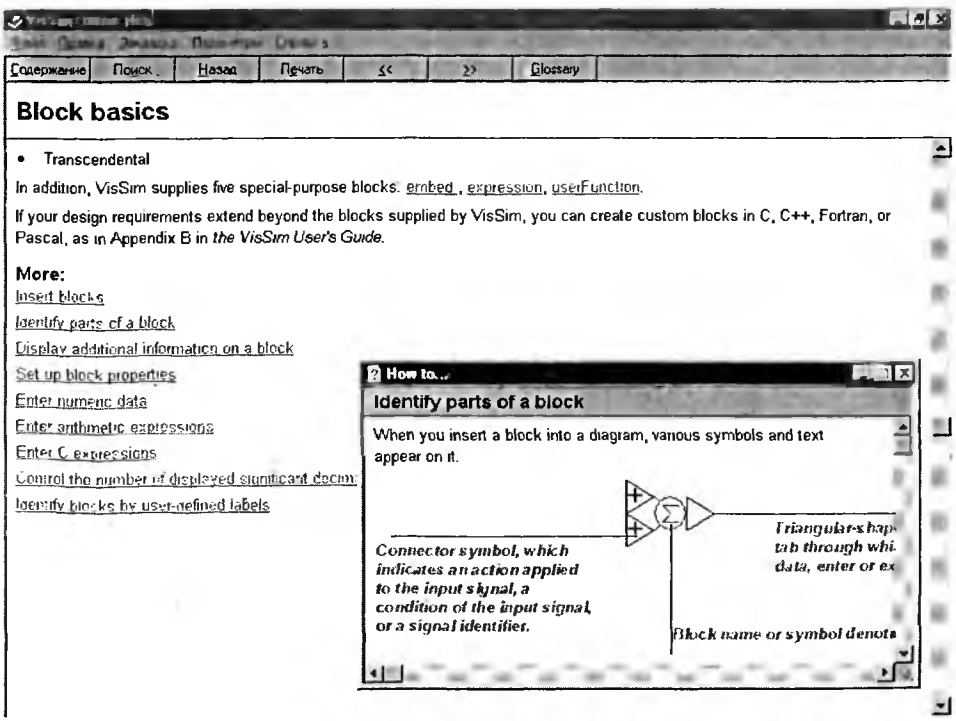


Рис. 2.44. Конец раздела справки по библиотеке блоков

Как уже отмечалось, окна свойств блоков имеют кнопку Help, которая также открывает окно справки по контексту — на этот раз по теме, относящейся к свойствам блока, его параметрам и типовым применениям.

2.10.3. Онлайновая справка

Команда Search... открывает стандартное окно *онлайновой справки*, показанное на рис. 2.45. Это окно имеет три вкладки:

1) Содержание — поиск по содержанию;
2) Указатель — поиск по алфавитному указателю;
3) Поиск — поиск по заданному слову или фразе.

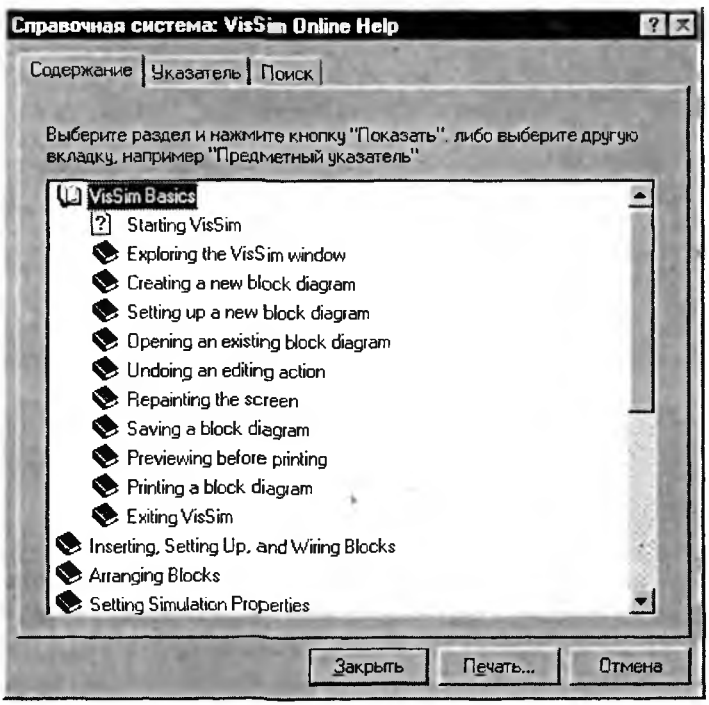


Рис. 2.45. Окно стандартной онлайновой справки с открытой вкладкой Указатель

Поскольку онлайновая справка используется практически во всех программах под операционную систему Windows, подробное описание методики работы с этим видом справки можно опустить. Окно справки стандартное, поэтому в случае работы с русифицированной операционной системой Windows некоторые надписи в окне будут на русском языке. Удивляться этому не стоит.

2.10.4. Справка по соединениям

Отметим, что необычная команда Wiring..., характерная именно для системы VisSim, просто открывает окно онлайновой справки с вкладкой Указатель и установкой в раздел, содержащий слово wiring (соединение) (рис. 2.46).

Рис. 2.46. Окно онлайновой справки по соединениям блоков

Работа с остальными разделами справки подобна описанной выше. Особо стоит отметить команду подменю справки Web Updates... Она открывает окно приложения Microsoft Internet Explorer (или иного доступного в данный момент интернет-браузера) и обеспечивает загрузку страницы обновления VisSim с сайта разработчика системы (см. раздел ниже).

2.11. VisSim в Интернете

2.11.1. Интернет-сайт разработчика VisSim

VisSim интенсивно развиваемая система. На сайте разработчика системы www.vissim.com (рис. 2.47) выставлена уже версия VisSim 5.0. Эту действующую в течение 60 дней версию можно скачать бесплатно. По истечении этого срока или при манипуляциях с календарем эта версия превращается в демонстрационную версию, в которой можно просматривать диаграммы, но нельзя их записывать на диск.

На этой странице можно получить доступ к разнообразной информации по программным продуктам корпорации Visual Solutions Incorporated, их обновлениям, документации, примерам применения и т. д. Особенно следует отметить пакеты расширения VisSim — Add Ons. После инсталляции таких пакетов в меню VisSim появляются новые позиции и возможности системы существенно расширяются. Один из таких пакетов расширения Analyze поставляется как отдельно, так и в составе описанных в данной книге версий

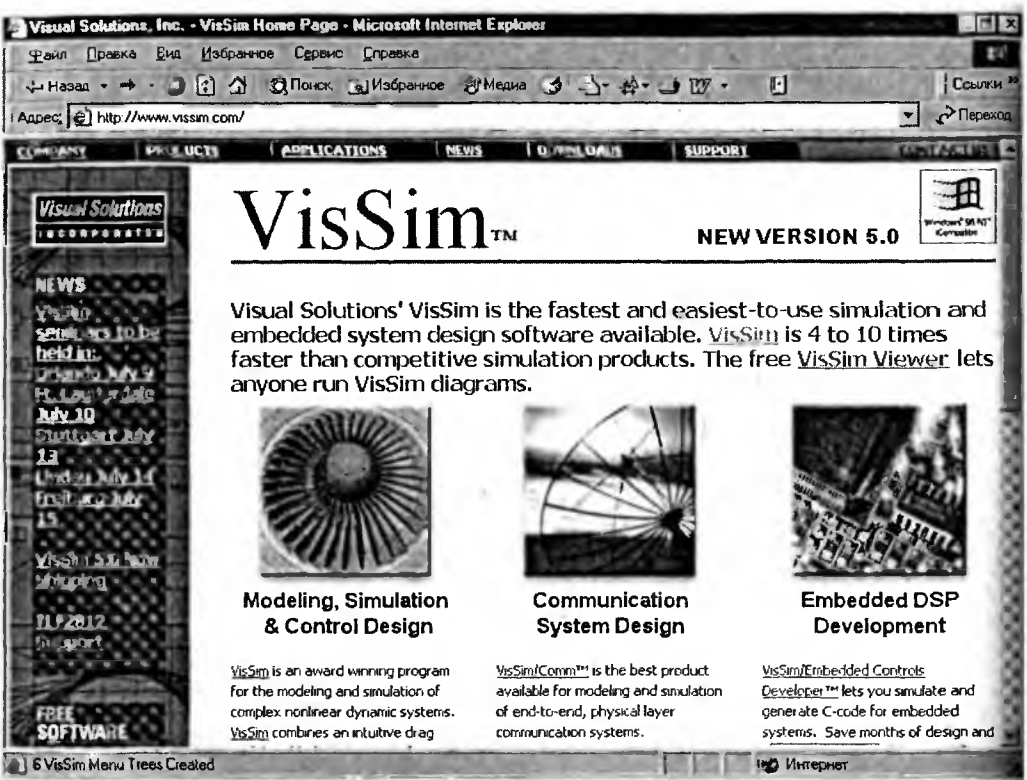


Рис. 2.47. Главная интернет-страница корпорации Visual Solutions Incorporated

системы VisSim. Если он установлен, то в меню появляется позиция Analyze. Пакет VisSim/Com с богатой библиотекой блоков для создания моделей коммуникационных устройств достаточно подробно описан в конце главы 6.

2.11.2. VisSim в России

VisSim достаточно известная в России система моделирования. Общедоступная версия VisSim 3.0, размещаемая в заархивированном виде даже на гибком диске, и перевод справки по этой системе на русский язык имеются на интернет-сайте vissim.nm.ru (его поддерживает Н. В. Климачев, Челябинск). Вид начальной страницы этого сайта показан на рис. 2.48.

На этом сайте приведена карта распространения системы VisSim в России (рис. 2.49). Города (увы, без названий), где VisSim применяется в учреждениях, помечены квадратиками. Нетрудно заметить, что область распространения системы простирается от Дальнего Востока до западных границ Российской Федерации.

На указанном сайте приведены также данные о многих вузах и университетах, уже использующих VisSim для учебного процесса по курсам, связанным с математическим моделированием. Однако в монографической, справочной и иной массовой литературе система VisSim пока не представлена. Этот пробел восполняет данная книга.

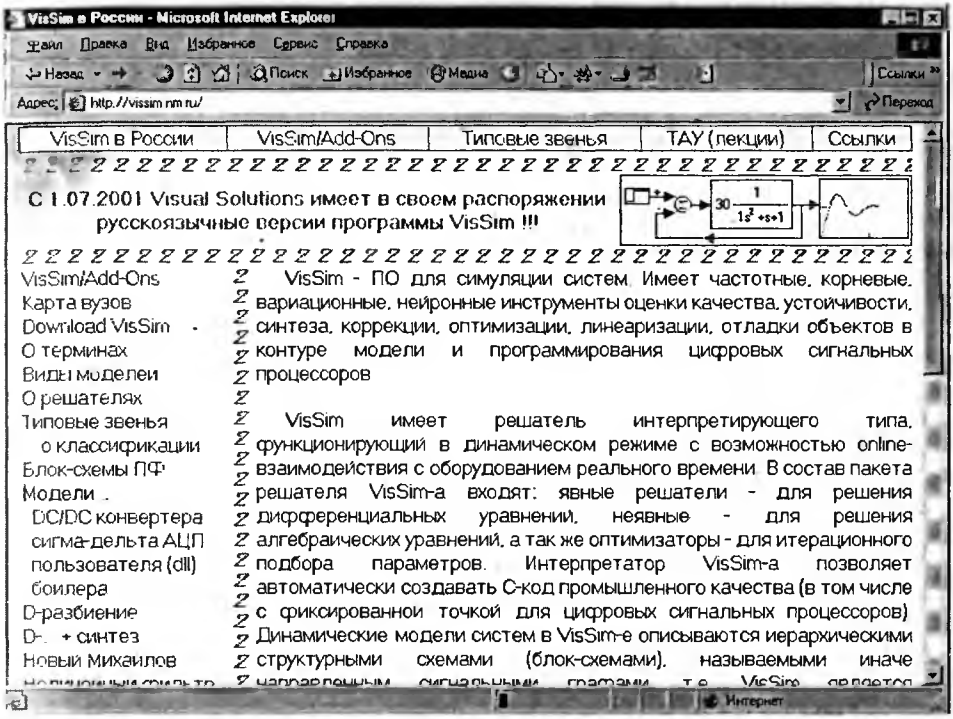


Рис. 2.48. Главная интернет-страница Российского сайта vissim.nm.ru

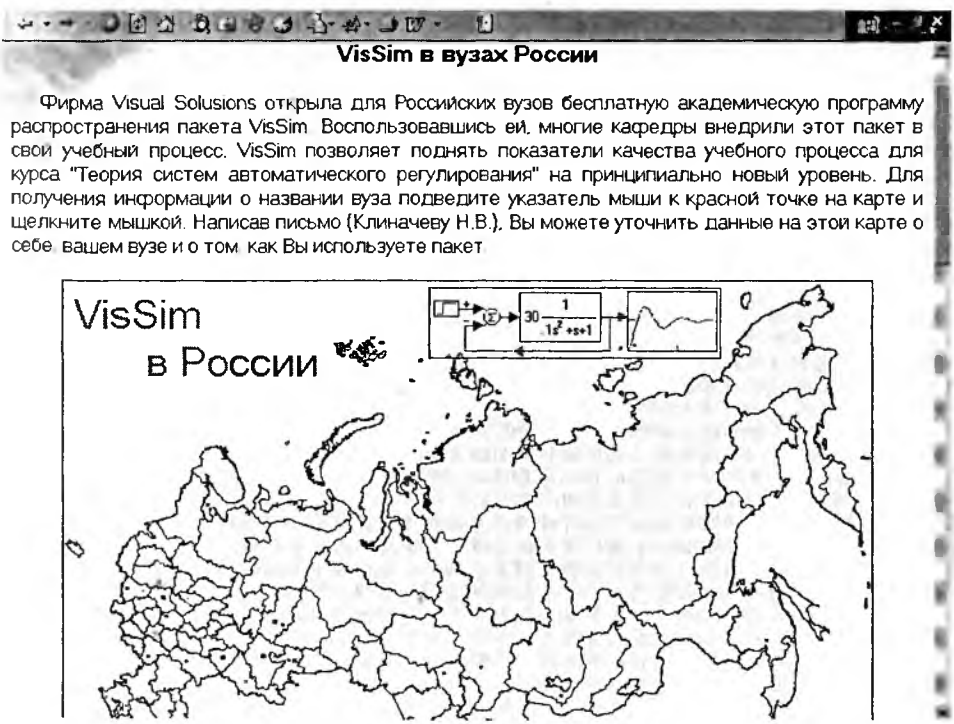


Рис. 2.49. Карта распространения системы VisSim в России

2.12. Версии VisSim 3.0 и 5.0

2.12.1. Версия VisSim 3.0

С интернет-сайта vissim.nm.ru можно скачать общедоступную (бесплатную) версию системы VisSim 3.0. Ее окно представлено на рис. 2.50. Как отмечает Н. В. Колмачев, эта версия была официально предоставлена российским пользователям разработчиком системы VisSim в качестве дара за активное участие в разработке системы и примеров ее применения.

Открытость версии VisSim 3.0 делает ее особенно привлекательной для наших образовательных учреждений, не имеющих достаточных средств для приобретения легальных программных продуктов. Нельзя не отметить, что VisSim 3.0 является вполне полноценной системой с немного урезанными по сравнению с VisSim 4.5 возможностями (нет позиции меню Tools и немного сокращен набор блоков). Но в сравнении с VisSim 4.5 Personal Edition у нее есть блок задания функции пользователя. Версия VisSim 3.0 позволяет решать практически любые задачи моделирования из курсов по моделированию, которые даются в большинстве наших вузов.

Самым главным недостатком VisSim 3.0 надо признать отсутствие блока связи с системой компьютерной математики Mathcad. В то же время средства интеграции с системой MATLAB присутствуют.

Несколько слов стоит сказать о так называемых «русских» версиях VisSim. По мнению автора данной книги, разумеется, в какой-то мере субъективного, русификация программ математического моделирования (как и других мате

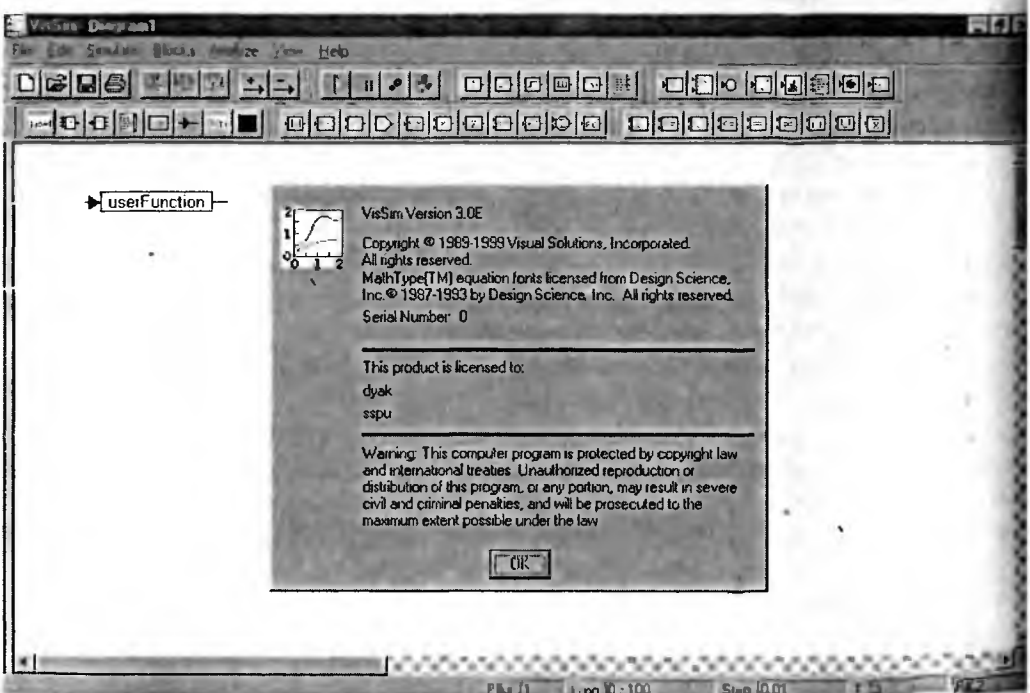


Рис. 2.50. Окно системы VisSim 3.0 и окно с данными о ней

матических программ) наносит больше вреда для их распространения и применения, чем пользы. Большинство понятий из области математического моделирования давно стало интернациональными и замена англоязычных команд и справки русскоязычными с довольно корявым переводом мало способствует пониманию пользователями деталей работы с такими программами.

2.12.2. Особенности новейшей версии VisSim 5

С интернет-сайта Visual Solutions Inc. можно также скачать новейшую (на момент написания данной книги) Trial-версию системы VisSim 5.0. Эта версия полностью работоспособна в течение 60 дней после ее установки, что нередко оказывается достаточным даже для проведения моделирования по нескольким достаточно серьезным проектам. Если система вам всерьез приглянулась, то надо связаться с Visual Solution Inc. на предмет условий ее дальнейшего использования, как правило, уже на коммерческой основе.

В новейшую версию VisSim 5 добавлены следующие возможности:

- новые блоки треугольного, прямоугольного и пилообразного сигналов;
- улучшенная вставка редактирования субблоков и их запись с помощью команды Save Embedded Files в позиции File меню;
- меню Pop-up в позиции Edit меню, обеспечивающее выполнение команд с блоками;
- новые команды автоматического соединения Auto Connect и разъединения Auto Disconnect в позиции Edit меню;
- команды выравнивания блоков слева, справа или сверху с улучшенной трассировкой соединений;
- новая команда Goto tags в позиции Edit меню;
- автоматическое конструирование таблиц с помощью блока dialogTable block;
- открытие сразу нескольких диаграмм и сравнение их друг с другом;
- новые блоки для задания матриц: диагональной diag(), единичной eye(), ones() и нулевой zeros();
- создание и отображение дисплеем комплексных данных (чисел);
- элементы событийно-управляемого моделирования;
- ускоренная оптимизация матриц;
- моделирование с заданием временного интервала или частоты;
- улучшенное задание параметров в блоке unitDelay;
- работа с компилятором Microsoft Visual C/C++ compiler (v5.0+);
- автоматическая трансляция файлов SIMULINK .mdl в файлы .vsm VisSim;
- интерактивное выполнение MATLAB команд, функций и m-файлов во время сессии моделирования VisSim;
- расширенные возможности мульти-графики;
- открытие отдельных окон графики во время сессии моделирования и возможность их закрытия командой Close в позиции File меню;
- генерация отчета по диаграмме;
- применимость команд Undo и Redo к субблокам разного уровня.

Из этих возможностей следует, безусловно, выделить те, которые связаны с дальнейшей интеграцией с другими программными средствами. Прежде всего это работа с компилятором языка C/C++, расширенные возможности в исполнении команд MATLAB и введение мастера для преобразования Simulink-моделей в VisSim-модели. Последняя возможность вызывает сомнения в ее целесообразности, поскольку библиотеки Simulink и VisSim существенно отличаются, а комплекс MATLAB+Simulink самодостаточен для решения задач моделирования. Тем не менее что есть, то есть.

2.12.3. Особенности интерфейса VisSim 5

Интерфейс версии VisSim 5 в принципе сохранил особенности интерфейса предшествующих версий (рис. 2.51). Из-за некоторого увеличения числа блоков в библиотеке панели инструментов под меню располагаются уже в трех ярусах, а не в двух. Заметно расширилось число команд в позиции Edit (Правка) меню, которое на рис. 2.51 представлено в открытом виде. Новое подменю выравнивания блоков в диаграммах Allign, также показанное на рис. 2.51 в открытом виде, содержит ряд очевидных команд выравнивания блоков диаграммы.

Некоторое изменение порядка позиций в меню к принципиальным отличиям не отнесешь, а вот появление новой позиции Windows бросается в глаза. Дело в том, что VisSim 5 стал уже полноценной многооконной системой, и теперь в него можно загружать множество диаграмм. Каждая диаграмма может занимать

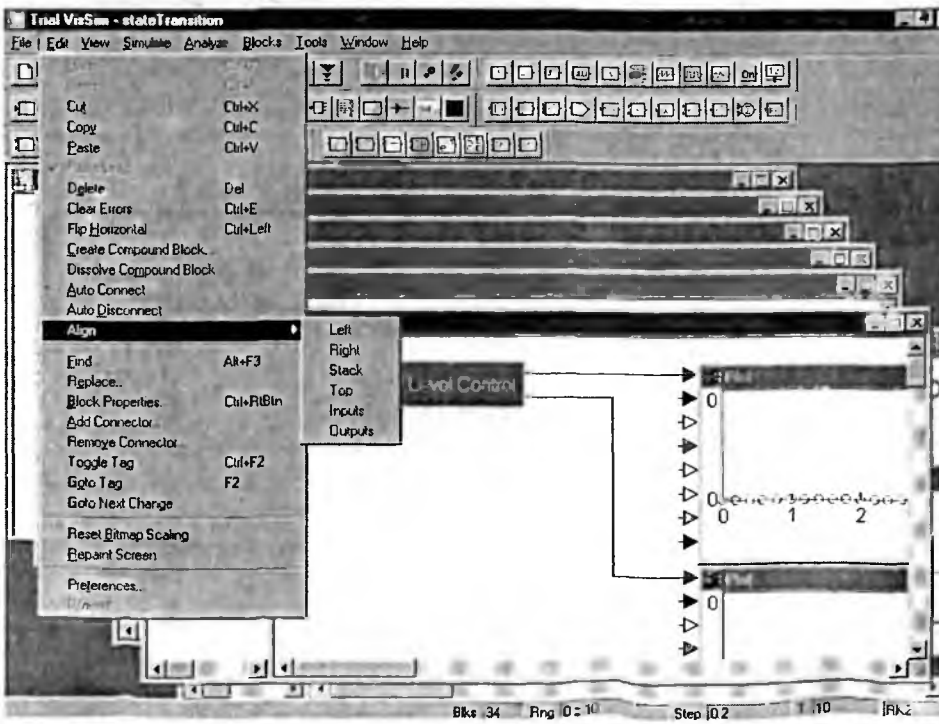


Рис. 2.51. Вид окна VisSim 5 с открытым меню правки Edit

свое окно. Позиция Windows обеспечивает различные варианты расположения окон. Так, на рис. 2.51 ряд окон показан в каскадном расположении (Cascade). Окна можно также расположить друг под другом (Tile). Команда New Window создает новое пустое окно. Каждое окно можно свернуть в пиктограмму, и для упорядоченного расположения их служит команда Arrange Icons. Под этими командами в позиции Windows имеется список загруженных приложений.

Многооконная работа VisSim 5 имеет по крайней мере два серьезных достоинства:

1) можно просматривать несколько диаграмм;

2) легко переносить блоки из одной диаграммы в другую, что существенно сокращает время подготовки диаграмм по имеющимся образцам.

В позиции Tools меню в подменю импорта данных появились команды загрузки данных из таблиц Look-Up Table... и Simulink Diagram.... Последняя команда свидетельствует об усилении роли интеграции с мощной системой MATLAB + Simulink. Есть в этой позиции и новая команда Difference Two Diagrams — сравнение двух диаграмм.

Как и в прежних версиях VisSim, набор позиций меню зависит от того, какие пакеты расширения установлены. Пакет Analyze является обязательным и порождает позицию меню Analyze. Корпорация Visual Solutions Inc. выпускает мощный пакет по моделированию коммуникационных устройств. Его версию класса Trial также можно скачать с интернет-сайта этой корпорации. После установки пакет порождает позицию меню Comm (рис. 2.52), в кото-

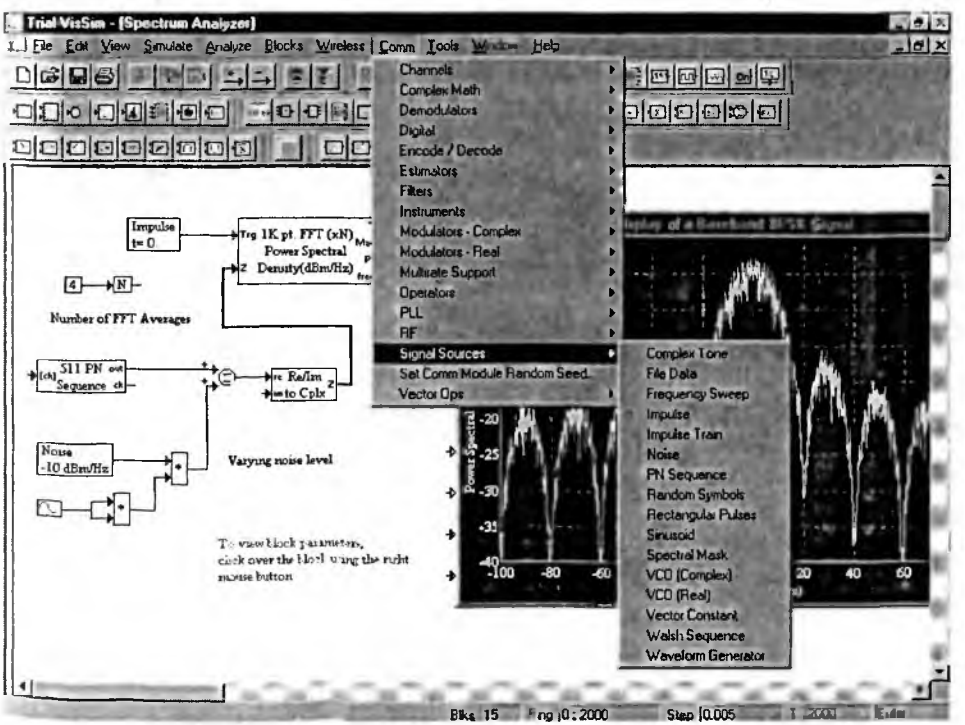


Рис. 2.52. Окно VisSim 5 с открытой позицией меню пакета расширений по моделированию коммуникационных систем и устройств

ром содержатся разделы мощной библиотеки по математической обработке сигналов и моделированию современных коммуникационных систем и устройств.

Возможности этого пакета мы рассмотрим в главе 6.

2.11.4. Сравнение диаграмм в VisSim 5

К интересным возможностям VisSim 5 относится сравнение двух диаграмм. Необходимость в этом связана с тем, что пользователь, создавая свои модели (диаграммы), часто создает ряд их копий. Порой они различаются лишь расположением блоков и мелкими деталями, что затрудняет сопоставление диаграмм. Теперь этой трудности нет.

Для сравнения некоторой заданной модели с другой моделью надо прежде всего загрузить сравниваемые модели. Затем в позиции Tools меню надо исполнить команду Difference Two Diagrams (Сравнить две диаграммы). Появится небольшое окно, показанное на рис. 2.53, в центре одной из диаграмм. В окошке имеется список загруженных диаграмм.

Из списка надо выбрать другую диаграмму, с которой надо провести сравнение. Нажав кнопку OK окна выбора диаграмм, можно получить окно, представленное на рис. 2.54. Окно в центре указывает на то, какие диаграммы сравниваются. Если блоки исходной диаграммы не встречаются в другой диаграмме, то они выделяются ярким красным цветом. Диаграмма сравнения может быть сохранена в файле.

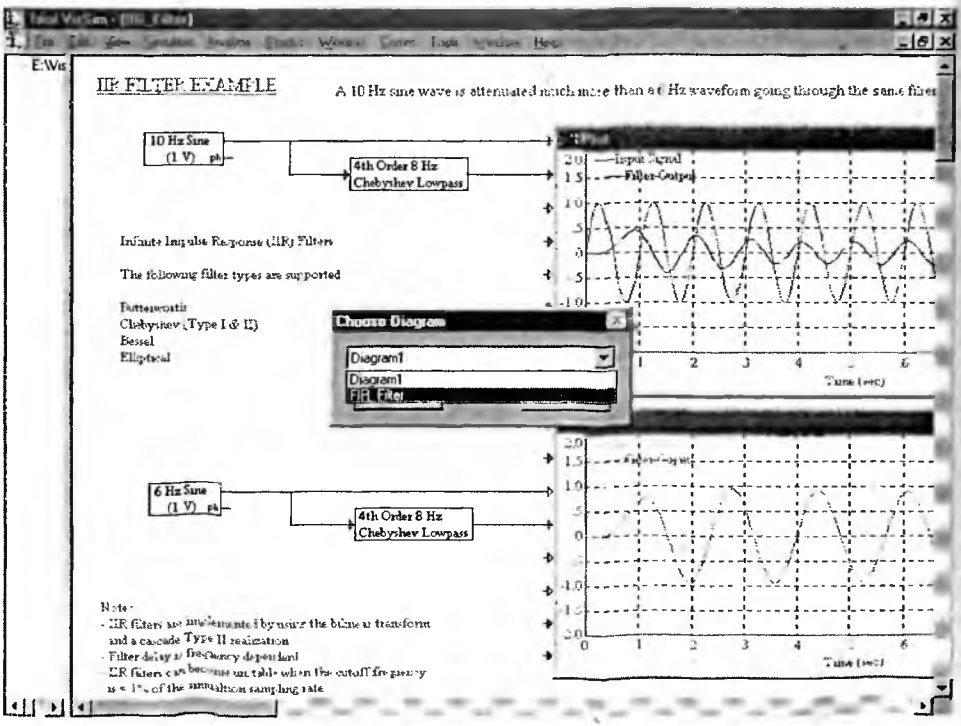


Рис. 2.53. Подготовка к сравнению двух моделей (диаграмм)

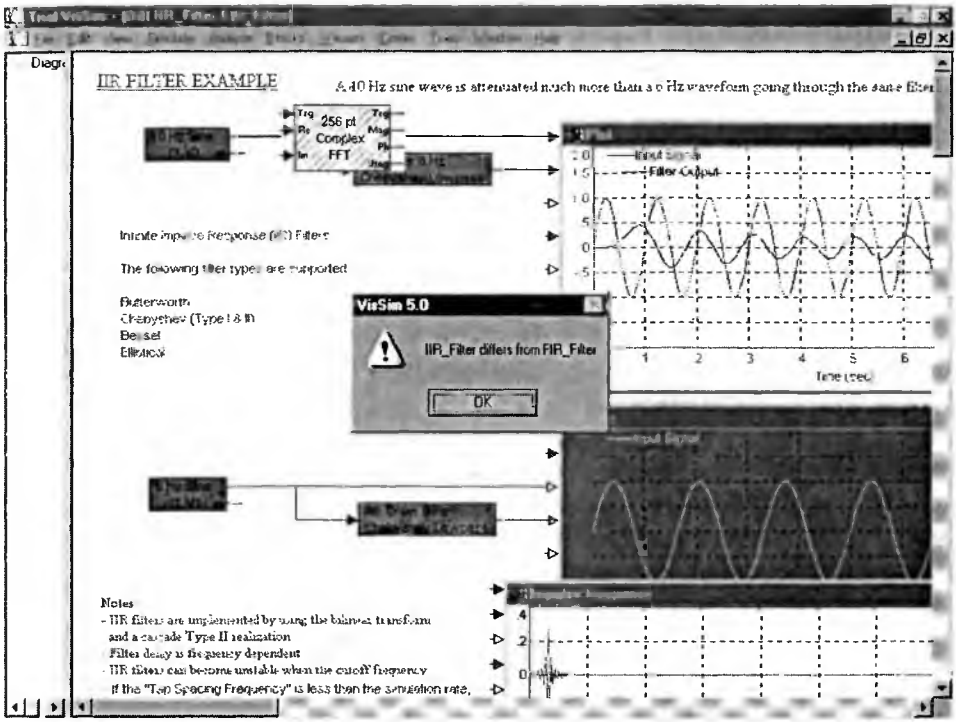


Рис. 2.54. Результат сравнения двух диаграмм

Безусловно, новая версия VisSim 5 имеет целый ряд привлекательных свойств. Однако для большинства пользователей широко распространенная версия VisSim 4.5 и даже миниатюрная и бесплатная VisSim 3 могут стать той синицей, о которой говорится в поговорке «лучше синица в руках, чем журавль в небесах». Напомним еще раз, что вы можете подержать в своих руках и красавца журавля (VisSim 5) целых два месяца и притом бесплатно!

Глава 3
Библиотека блоков и работа с ними

Эта одна из центральных и самых больших глав книги содержит исчерпывающую информацию по всем блокам основной библиотеки VisSim (их около сотни). Для каждого блока приводится его вид, окно свойств, список устанавливаемых параметров и, главное, простой и понятный пример применения. Освоение этих примеров необходимо начинающему пользователю системой VisSim, опытные пользователи могут пользоваться материалами этой главы выборочно. Для удобства работы с этой главой названия блоков вынесены в заголовки.

3.1. Раздел библиотеки Annotations

3.1.1. Пассивная фоновая панель bezel

Блок bezel служит для создания прямоугольной фоновой панели, которая выводится как бы позади графического изображения модели или ее фрагментов. Панель может содержать рисунок из файла типа BMP или просто иметь заданный цветной фон. Будучи наложенной на изображение того или иного блока или части модели, такая панель позволяет создавать очень наглядные и законченные графические представления моделей. Если режим Display mode не включен, то панель обрамляется рамкой и может меняться в размерах и перемещаться с помощью мыши. Если режим Display mode включен, то панель фиксируется и рамка исчезает.

В окне свойств панели, показанном на рис. 3.1, можно задать или изображение из файла или цвет панели. Для задания изображения надо активизировать кнопку Image (Изображение) и в появившемся стандартном окне загрузки файлов найти нужный файл. В окошке у кнопки можно наблюдать изображение, которое будет представлено в панели.

Если в области Color (Цвет) панели установить птичку у опции Use Solid Color (Заполнить однородным цветом), то можно задать заливку панели однородным цветом (рис. 3.2). В этом случае активизации кнопки Select Color... (Выбор цвета) появляется панель выбора цвета, показанная на рис. 3.2. В ней можно задать нужный цвет панели.

К сожалению, совместное применение фона в виде рисунка и сплошного цвета не предусмотрено.

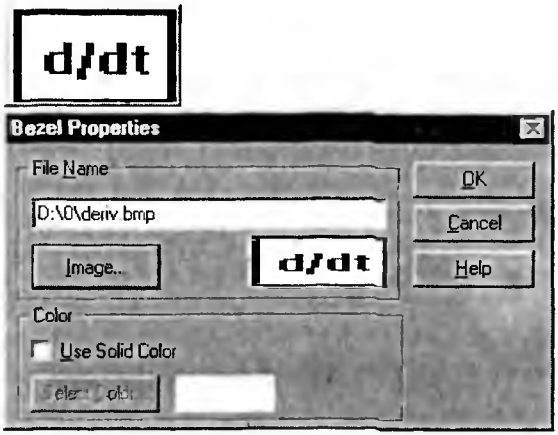


Рис. 3.1. Блок bezel (сверху) и панель его свойств (снизу) при задании изображения в панели

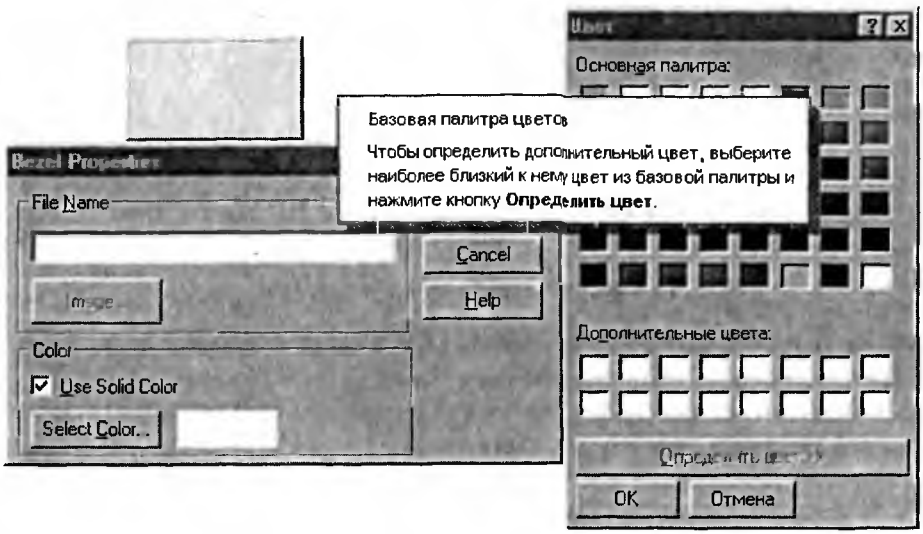


Рис. 3.2. Блок bezel (сверху) и панель его свойств (снизу) при задании цветового фона панели

3.1.2. Блок текстовых комментариев comment

Блок ввода текстовых комментариев comment создает прямоугольную область, в которую можно ввести текстовые комментарии (рис. 3.3). Этот блок не имеет окна свойств. Вводимый текст может редактироваться посредством типовых операций строчного редактирования. В ходе его возможно применение и стандартных операций работы с буфером промежуточного хранения Windows.

Для задания текстовых комментариев повышенного качества можно использовать текстовый процессор Word. При этом могут испо-

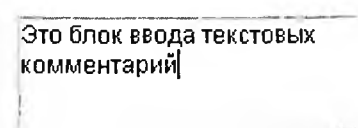


Рис. 3.3. Блок задания текстовых комментариев

льзоваться присущие Word возможности форматирования. Однако, чтобы их сохранить в окне текстовых комментарий VisSim, нужно задать опцию Use Rich Text Format (Использовать Rich текстовый формат) в окне глобальных настроек Preferences, вывод которого предусмотрен в позиции Edit меню.

3.1.3. Блок вывода даты

Для вывода текущей даты (дня недели, месяца и года) служит блок Data. Его вид показан на рис. 3.4.

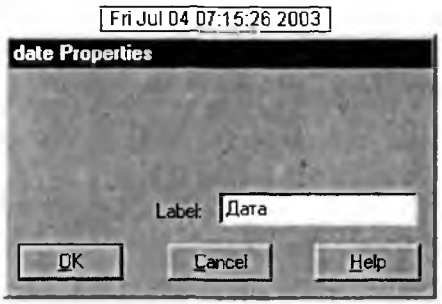


Рис. 3.4. Блок вывода даты и окно его свойств

Окно свойств этого блока содержит только установку метки блока. В дальнейшем такие окна показываться и описываться не будут.

3.1.4. Блок метки label

Блок метки служит для установки в окне моделей небольших текстовых комментариев (рис. 3.5). Изначально таким комментарием является слово «label», но его можно сменить на любой другой комментарий с помощью окна свойств этого блока.

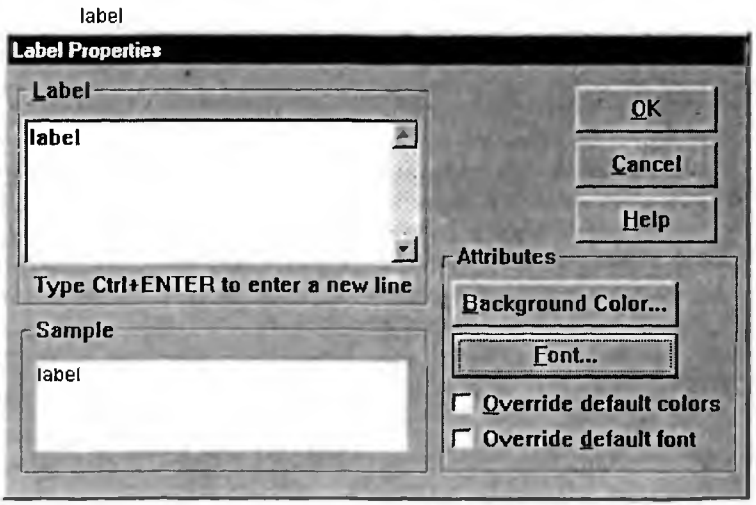


Рис. 3.5. Блок задания метки и окно установки его параметров

Для замены метки достаточно вывести окно установки параметров этого блока. Оно показано на рис. 3.5 под блоком метки. В окне Label можно задать новую надпись и просмотреть ее в окне Sample (Макет). Можно задать опции задания цвета фона и выбора желаемого шрифта. Первая опция с помощью кнопки Background Color... (Цвет фона) выводит окно выбора цвета, подобное приведенному на рис. 3.2, а вторая с помощью кнопки Font... (Шрифт) выводит стандартное окно Windows для выбора нужного набора символов и их стиля.

3.1.5. Блок превращения скалярных величин в вектор scalarToVec

Блок превращения скалярных величин в вектор scalarToVec служит для объединения нескольких скалярных величин в вектор (рис. 3.6). Из приведенного рисунка видно, что в блоках констант на входе блока scalarToVec могут использоваться выражения, а не только одиночные численные значения. Шина, передающая вектор, отображается жирной линией. Окно установки параметров блока scalarToVec позволяет задавать число строк Rows и столбцов Cols. Таким образом, строго говоря, блок преобразует набор скалярных величин в матрицу.

Здесь полезно отметить, что есть оперативный способ наблюдать содержимое вектора (матрицы) на выходе блока scalarToVec. Для этого надо поместить курсор мыши на выход блока и дождаться его превращения в стрелку ↑. Если после этого нажать правую клавишу мыши, то появится бирка с содержимым вектора (рис. 3.7).

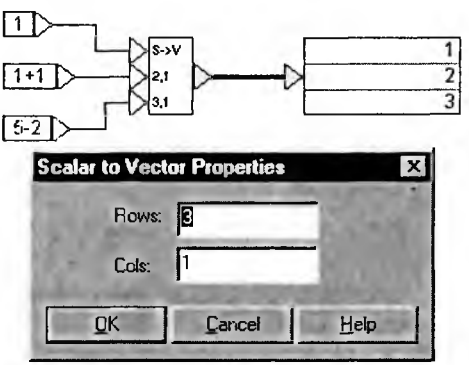


Рис. 3.6. Блок превращения скалярных величин в вектор scalarToVec и окно его свойств

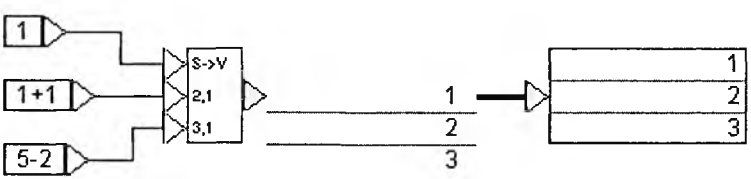


Рис. 3.7. Контроль за содержимым вектора (матрицы) на выходе блока scalarToVec

Сборка ряда сигналов в одну шину повышает наглядность моделей и упрощает их вид. Кроме того, такие сигналы могут использоваться в векторных и матричных операциях. Для преобразования векторного (или матричного) сигнала в набор скалярных сигналов можно использовать блок vecToScalar.

3.1.6. Блок задания переменных variable

Для задания переменных служит блок variable. Этот блок имеет вход, на который подается значение переменной, которое присваивается ей, а также выход. Переменные служат удобным средством осуществления «беспроводной» связи между блоками, что иллюстрирует рис. 3.8. В пределах ветви иерархии или уровня модели переменные могут использовать механизм локализации.

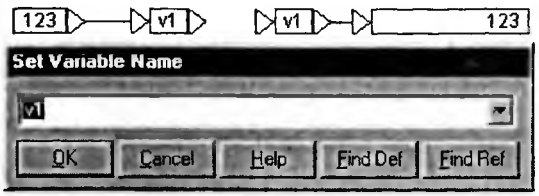


Рис. 3.8. Применение блоков задания переменных

3.1.7. Блок превращения вектора в скалярные величины vecToScalar

Для превращения вектора в скалярные величины служит блок vecToScalar. Его вид и применение показаны на рис. 3.9.

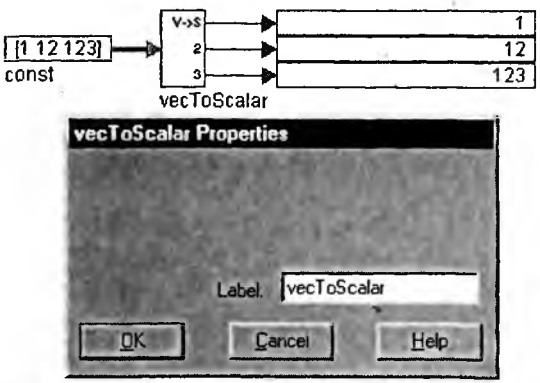


Рис. 3.9. Применение блока преобразования вектора в скалярные величины и окно его свойств

Назначение этого блока вполне очевидно — блок может использоваться для выделения отдельных элементов массива.

3.1.8. Блок фиксации соединения wirePositioner

Блок фиксации провода wirePositioner позволяет фиксировать положение соединения, что часто упрощает построение моделей. Например, на рис. 3.10 сверху построена модель системы с обратной связью без фиксации соединения, создающего обратную связь, а снизу — модель системы, содержащая блок фиксации этого соединения (сам блок представлен снизу модели).

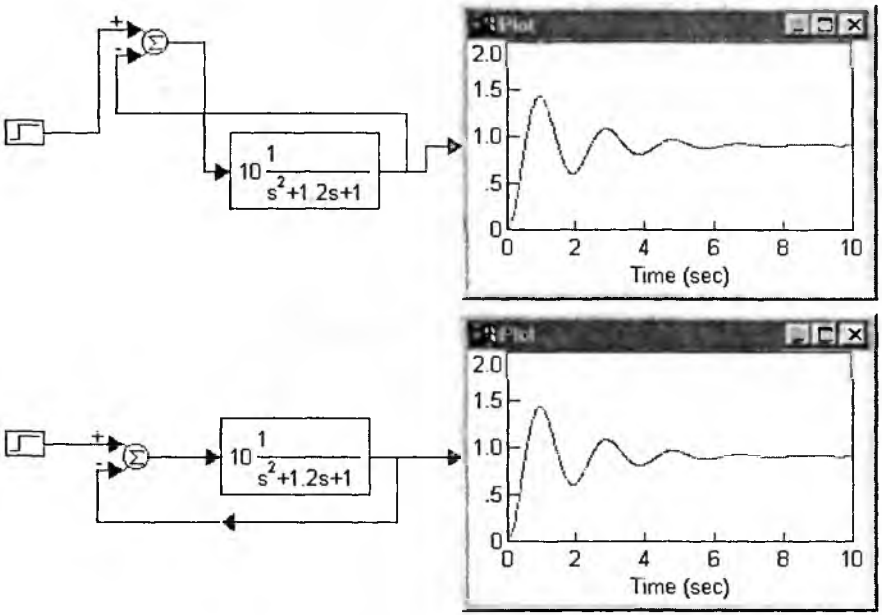


Рис. 3.10. Построение модели системы с обратной связью без применения блока фиксации соединения (сверху) и с таким блоком (снизу)

Применение блока фиксации соединения позволяет удобно располагать блоки диаграммы, не заботясь о прокладке соединений. Однако чрезмерно увлекаться применением этого блока не стоит, поскольку его изображение загромождает диаграмму.

3.2. Разделы библиотеки по математическим и логическим операциям

3.2.1. Блоки операций 1/X, -X, * и /

Блоки простейших арифметических операций выполняют следующие операции:

- 1/X — вычисление обратного значения;
- -X — смена знака;
- * умножение двух аргументов;
- / — деление двух аргументов (верхнего на нижнее).

Обозначение каждого этих блоков совпадает с обозначением указанной выше операции. Применение этих блоков для операций со скалярным аргументом представлено на рис. 3.11. Блоки могут выполнять и операции с векторами и матрицами. Ряд примеров применения этих блоков можно найти в поставке примеров для системы VisSim.

Блок деления можно использовать для деления массивов одинаковых размеров или массива на скаляр. В последнем случае реализуется почленное де-

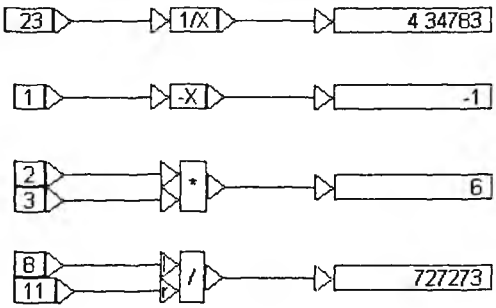


Рис. 3.11. Примеры применения блоков операций 1/X, -X, * и /

лений. Учтите, однако, что применять оператор деления для вычисления обратной матрицы нельзя. При делении на ноль появляется сообщение об этом, устанавливается флаг ошибки, и блок закрашивается красным цветом.

3.2.2. Блок вычисления абсолютного значения abs

Блок вычисления абсолютного значения abs для аргумента x вычисляет $|x|$ или $y = x$ при $x >= 0$ и $y = -x$ при $x < 0$. Пожалуй, самым наглядным применением этого блока является моделирование работы идеализированного двухполупериодного выпрямителя переменного синусоидального напряжения (рис. 3.12).

Операция abs, как и многие другие математические операции, может использоваться при аргументах типа векторов и матриц. Пример такого применения блока abs представлен на рис. 3.13.

Обратите внимание на то, что во втором примере блок scalarToVec использован для преобразования скалярных величин в матрицу, отображаемую как двумерный массив.

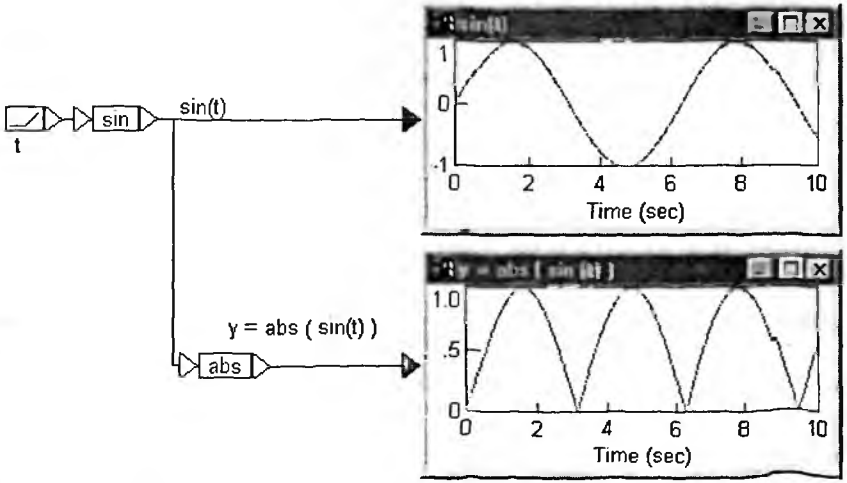


Рис. 3.12. Применение блока abs для имитации работы двухполупериодного выпрямителя

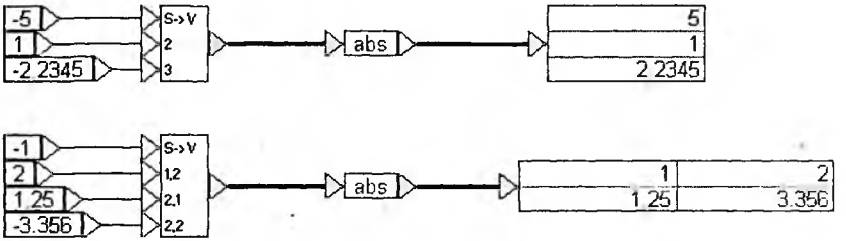


Рис. 3.13. Применение блока abs для векторного и матричного аргументов

3.2.3. Блок контроля знака sign

Блок контроля знака sign при скалярном аргумента x возвращает 1, если $x > 0$, –1, если $x < 1$, и 0, если $x = 0$. Примеры применения блока представлены на рис. 3.14.

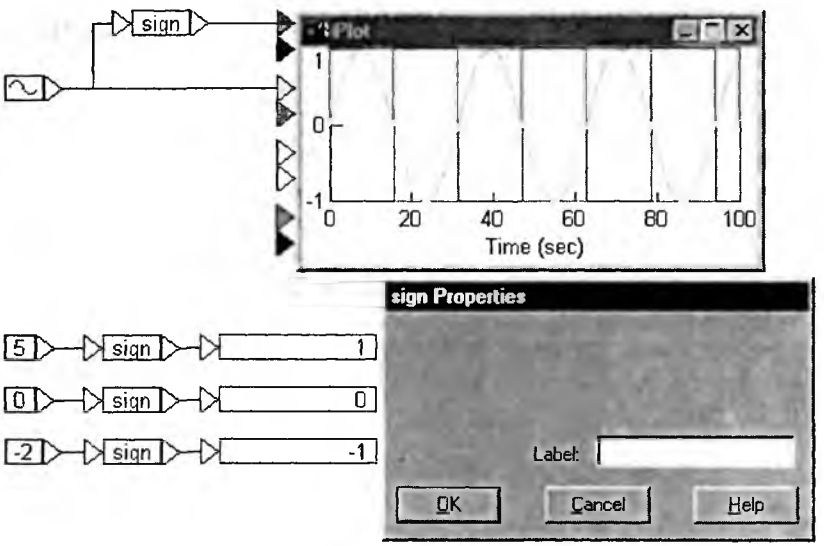


Рис. 3.14. Примеры применения блока .sign

Операция sign преобразует синусоидальный сигнал в симметричные прямоугольные импульсы, именуемые меандром, что видно из рис. 3.14. Как и в предшествующих блоках, параметров у блока sing нет. Это видно из окна свойств этого блока, показанного на рис. 3.14. Возможно задание только метки Label.

3.2.4. Блок преобразования размерных величин unitConversion

Блок преобразования размерных величин unitConversion служит для преобразования данных, представленных в размерных величинах из одной системы единиц в другую, например температуры, выраженной в градусах Цельсия, в температур, выраженную в градусах Кельвина или Фаренгейта. Пример та-

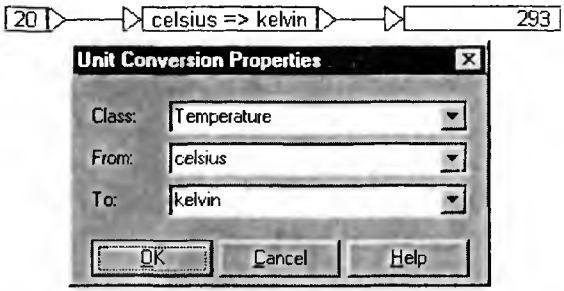


Рис. 3.15. Пример преобразования температуры из градусов Цельсия в градусы Кельвина

Окно свойств блока unitConversion имеет три списка:

1) Class — выбор типа преобразуемых размерных величин;
2) From — выбор величины, с которой осуществляется преобразование;
3) To — выбор величины, в которую осуществляется преобразование.

В обширный список преобразуемых величин (рис. 3.15) входят: длина, площадь, объем, величина потока, давление, скорость, ускорение, масса, сила, мощность, угловые величины, скорость вращения, момент, инерция вращения, температура, энергия, заряд, емкость, индуктивность, сопротивление, ток, напряжение, магнитный поток и др. Преобразуемые величины отображаются в блоке.

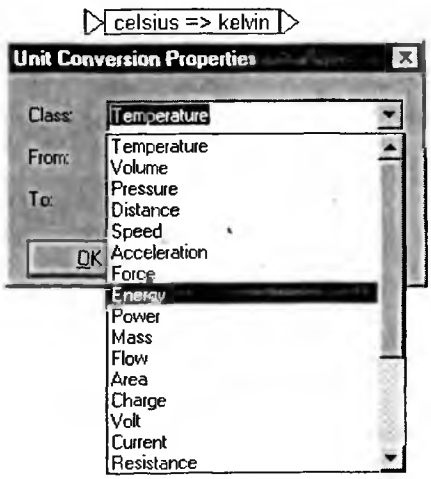


Рис. 3.16 Окно свойств блока unitConversion с открытым списком размерных величин

3.2.5. Блок преобразования типов данных convert

Блок преобразования типов данных convert преобразует один тип данных на своем входе в другой тип данных на выходе. Примеры таких преобразований представлены на рис. 3.17. В левой части рисунка показано окн

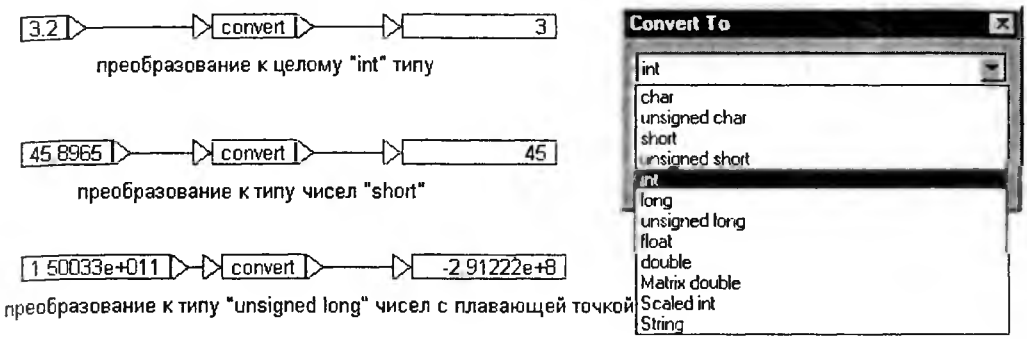


Рис. 3.17. Применение блоков преобразования типов данных

свойств блока с раскрытым списком типов, к которым осуществляются преобразования.

Осуществляются преобразования в следующие типа данных: char, unsigned char, short, unsigned short, int, long, unsigned long, float и double. Все они достаточно известны.

3.2.6. Блок масштабирования gain

Блок масштабирования gain осуществляет операцию y = gain*x. Величина gain является коэффициентом передачи. Этот блок нельзя отождествлять с усилителем, поскольку он безынерционный и его нельзя охватывать отрицательной обратной связью. На рис. 3.18 представлено применение блока gain для получения синусоидального сигнала удвоенной (по сравнению с сигналом на входе) амплитудой.

Окно свойств блока предусматривает установку параметры gain.

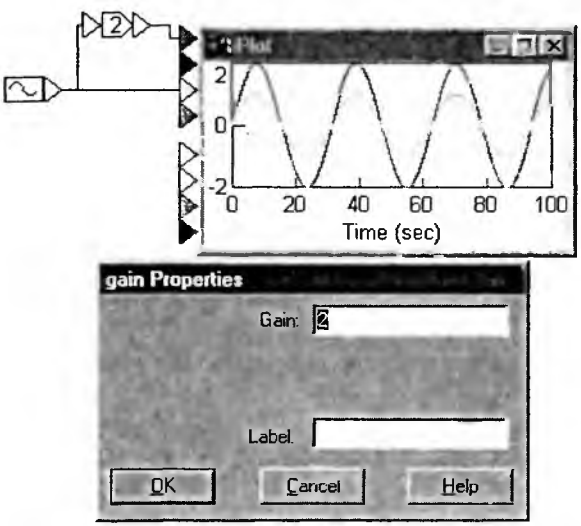


Рис. 3.18. Применение блока gain для создания синусоидального сигнала двойной амплитуды

3.2.7. Блок возведения в степень pow

Блок возведения в степень pow служит для выполнения операции $y = x^{pow}$. На рис. 3.19 показано возведение синусоидального сигнала в третью степень. В окне свойств блока кроме метки предусмотрено задание параметра pow.

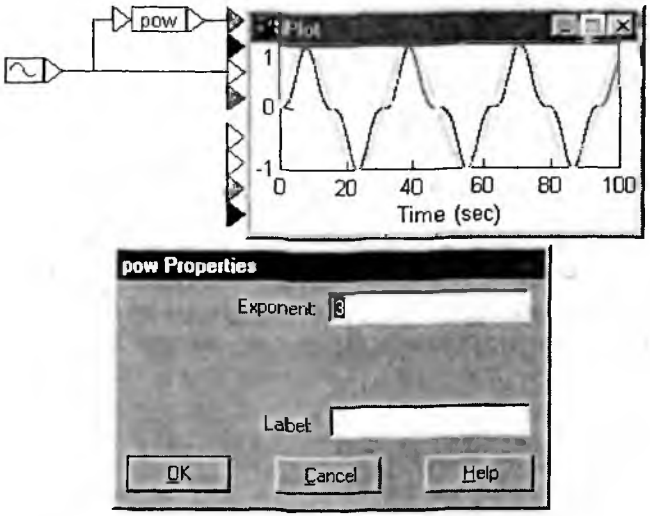


Рис. 3.19. Возведение синусоидального сигнала в третью степень с помощью блока pow

3.2.8. Блок суммирования/вычитания summingJunctions

Блок суммирования summingJunctions служит для выполнения операции $y = x1 + x2$, где x1 и x2 — сигналы на входах блока. Рисунок 3.20 показывает сложение для двух скалярных величин — чисел 2 и 3, а также сложение нарастающего по линейному закону сигнала с синусоидальным сигналов.

Этот блок может использоваться с векторными и матричными данными. По любому входу можно задать операцию вычитания. Для этого при нажатой клавише Ctrl подведите курсор мыши к нужному входу и в момент смены об-

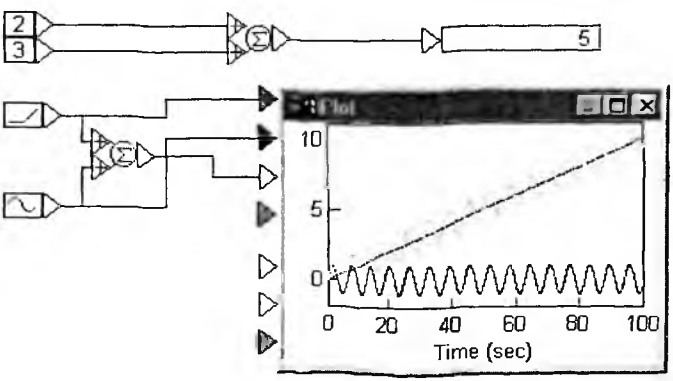


Рис. 3.20. Примеры применения блока summingJunctions

раза курсора на стрелку ↑ нажмите правую кнопку мыши. Знак + у входа сменится на знак −. Аналогичным образом можно осуществить и обратную замену. Данный блок часто используется для осуществления положительной или отрицательной обратной связи в моделируемых системах.

3.2.9. Трансцендентные функции раздела Transcendental

Целый раздел библиотеки Transcendental содержит блоки трансцендентных функций:

- acos — тригонометрическая функция арккосинуса;
- asin — тригонометрическая функция арксинуса;
- atan2 — тригонометрическая функция арктангенса;
- bessel — функция Бесселя;
- cos — тригонометрическая функция косинуса;
- cosh — функция гиперболического косинуса;
- exp — функция вычисления экспоненциального значения;
- ln — функция вычисления натурального логарифма;
- log10 — функция вычисления логарифма с основанием 10;
- sin — тригонометрическая функция синуса;
- sinh — функция гиперболического синуса;
- sqrt — функция вычисления квадратного корня;
- tan — тригонометрическая функция тангенса;
- tanh — функция гиперболического тангенса.

Все эти блоки, за исключением atan2, имеют один вход и один выход и представляются прямоугольником с названием функции в нем. Блок atan2 вычисляет значение atan(x,y) арктангенса угла, который задается положением радиус-вектора, конечная точка которого имеет координаты (x,y), а начальная (0,0). Примеры применения части из описанных функций представлены на рис. 3.21.

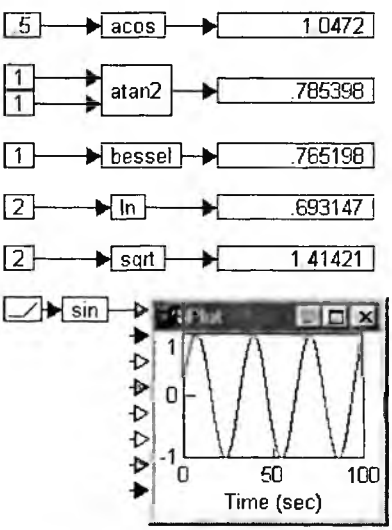


Рис. 3.21. Примеры применения блоков трансцендентных функций

3.3. Блок задания выражения expression

Довольно часто надо задать какое-либо выражение, например x1*sin(x2). Для этого служит блок expression. Он имеет один вход и один выход, и с помощью окна свойств в него можно ввести заданное выражение. Если оно содержит более одного аргумента, то число входов блока можно увеличить, используя команду Add connector в позиции Edit меню. Переменные в этом случае обозначаются как $1, S2 и т. д. Пример применения блока expression представлен на рис. 3.22.

Если в записываемом выражении есть ошибки, то ввести такое выражение не удастся, и в поле Parse Error появятся сообщения об ошибках. Пример этого при попытке ввести выражение a*sin(b) показан на рис. 3.23. Ошибка в данном случае в том, что вводятся неопределенные переменные a и b, тогда как допустимы переменные $1 и $2.

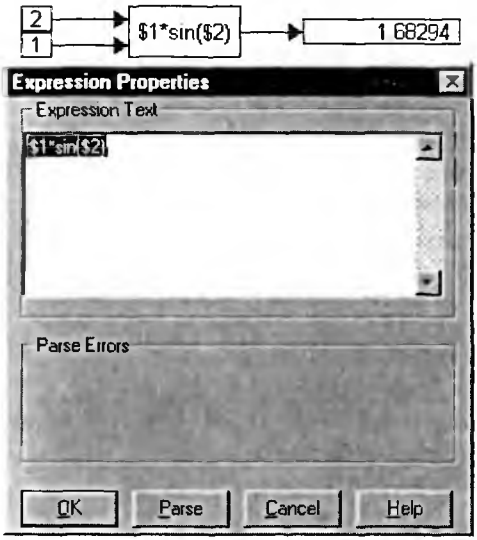


Рис. 3.22. Применение блока expression для вычисления выражения $1*sin($2)

Рис. 3.23. Пример неправильного ввода выражения в блок expression

3.4. Блоки логических операций и функций раздела Boolean

3.4.1. Блоки логических операций

Библиотека VisSim имеет набор блоков для выполнения стандартных логических операций сравнения:

- > — больше;
- < — меньше;
- >= — больше или равно;
- <= — меньше или равно;
- == — равно;
- =! — не равно.

Все эти блоки двухвходовые. На входы могут подаваться числа любого типа. Выход блоков один, и он возвращает значение 1 (логическая единица), если условие сравнения выполняется, и 0 (логический ноль), если условие сравнения не выполняется. Рисунок 3.24 показывает применение блока >. Аналогичным образом могут использоваться и другие блоки сравнения.

Интересно, что эти блоки не имеют окна свойств. Вместо него выводится список блоков, что позволяет быстро заменить один тип блока сравнения на другой. Этот список виден на рис. 3.24.

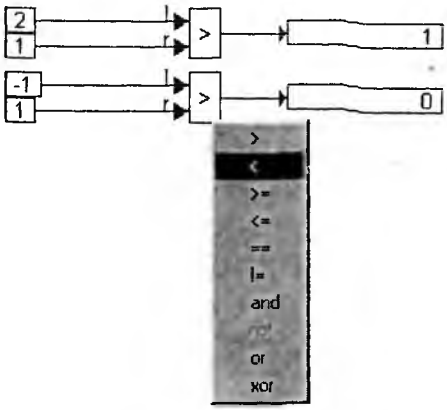


Рис. 3.24. Применение блока сравнения > (больше)

3.4.2. Блоки логических функций

В разделе библиотеки Boolean есть также блоки для реализации логических функций:

- and — функция логического сложения И;
- not — функция логического отрицания;
- or — функция логического умножения ИЛИ;
- xor — функция исключающая ИЛИ.

Блок функция not логического отрицания одновходовый. Если на его входе действует 1, то на выходе — 0, а если на входе действует 0, то на выходе будет 1. Блоки остальных функций двухвходовые, но число их входов можно увеличить. Примеры применения блоков логических функций представлены на рис. 3.25.

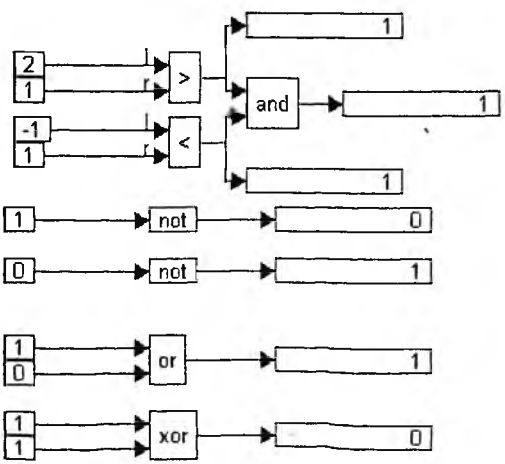


Рис. 3.25. Примеры применение блоков логических функций

3.5. Интерфейсные блоки

3.5.1. Блок порта ввода rt-DataIn

Блок rt-DataIn осуществляет чтение и передачу в модель сигнала, поступающего из аппаратных средств сопряжения компьютера с внешними объектами. Настройка блока на порты конкретного аппаратного устройства осуществляется посредствам команды меню Real Time Comfig... Должен быть задан режим реального времени для процесса симуляции.

3.5.2. Блок порта вывода rt-DataOut

Блок rt-DataOut осуществляет запись поступающего на его вход сигнала в аппаратные средства сопряжения компьютера с внешними объектами. Настройка блока на порты конкретного аппаратного устройства осуществляется посредствам команды меню Real Time Config... в позиции File меню.

Блоки rt-DataIn и rt-DataOut недоступны, если в компьютере не установлены какие-либо расширения для работы в реальном масштабе времени, например VisSim/Real-Time, VisSim/Real-TimePRO или VisSim DACQ.

3.5.3. Блоки для работы с ActiveX

В версии VisSim появились новые блоки для работы с динамическими объектами класса ActiveX. Введены три относящихся к этим объектам блока:

1) ActiveX read — считывание;

2) ActiveX write — запись;

3) About VisSim ActiveX Interface — вывод окна с данными о версии интерфейса с объектами ActiveX (версия 1.0).

Простой пример применения блоков записи и считывания ActiveX показан на рис. 3.26. При пуске можно наблюдать запись в объект ActiveX write с именем level текущего уровня синусоидального сигнала и тут же повторение на выходе блока ActiveX read.

Окна свойств параметров этих блоков имеют список для выбора объектов по их имени Name и поле с выведенным значением Value.

3.5.4. Блоки интерфейса DDE (dynamic data exchange)

Для организации динамического обмена данными (dynamic data exchange) в разделе библиотеки DDE имеется три блока. Их назначение описано ниже.

Блок DDEreceive (приемник) позволяет в процессе моделирования организовать динамический прием данных в блок-схему VisSim из другого приложения Windows. Здесь приложение будет сервером, а VisSim — клиентом.

Блок DDEsend (передатчик) позволяет в процессе моделирования организовать динамическую передачу данных из блок-схемы VisSim в другое приложение Windows. Здесь VisSim будет сервером, приложение — клиентом.

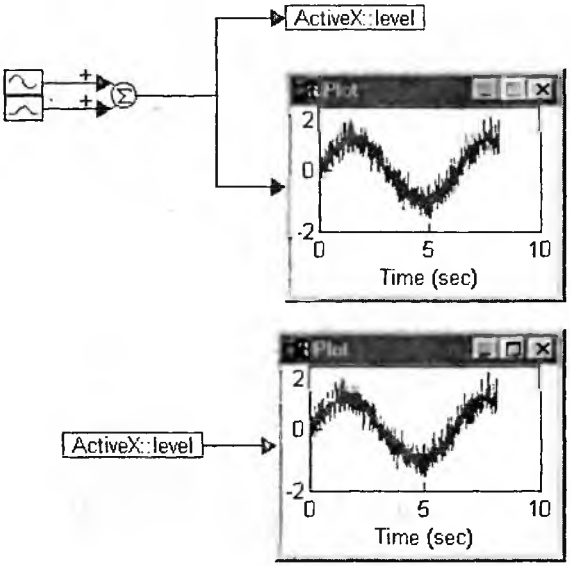


Рис. 3.26. Пример применения блоков ActiveX write и ActiveX read

Блок DDE-интерфейс позволяет организовать в процессе моделирования двусторонний динамический обмен данными между VisSim и другим приложением Windows.

3.6. Интегрирующие блоки

3.6.1. Блок идеального интегратора 1/S

Моделирование очень часто требует реализации функции интегрирования, которая выражается следующей формулой:

$$y = y(t_{start}) + \int_{t_{start}}^{t_{end}} x(t)dt.$$

Эту функцию реализует блок идеального интегратора 1/S. Его обозначения исходят из операторного выражения для операции интегрирования. Интегрирование задается от начального момента времени t_{start} до конечного t_{end}. Таким образом, блок вычисляет численное значение определенного интеграла при начальном значении (условии) $y(t_{start})$. Идеальность блока заключается в отсутствии ограничений на значение вычисленного интеграла y, максимальное и минимальное значение которого могут ограничиваться только разрядной сеткой компьютера.

Применение блока идеального интегратора иллюстрирует рис. 3.27. Здесь даны два случая применения блока: интегрирование константы при начальном значении 0,2 и интегрирование синусоиды при нулевом начальном значении. В первом случае интегратор дает линейно нарастающее со значения 0,2 значение входного сигнала, во втором случае — косинусоидальный сигнал.

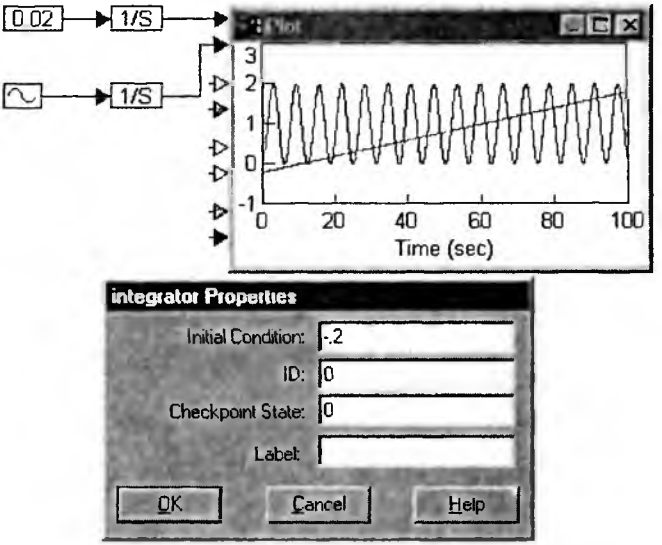


Рис. 3.27. Применение блока идеального интегратора

Здесь важно отметить, что интегрирование задается численным методом в соответствии с установками на вкладке Integration Method окна свойств моделирования (команда Simulation Properties... в позиции Simulation меню). Могут быть заданы следующие методы интегрирования: Эйлера, трапеций, Рунге—Кутта 2-го порядка, Рунге—Кутта 4-го порядка, адаптивный метод Рунге—Кутта 5-го порядка, адаптивный метод Булирша—Стоера или обратный Эйлера (для интегрирования жестких систем уравнений). По умолчанию задан метод Рунге—Кутта 2-го порядка, обеспечивающий высокую скорость интегрирования и приемлемую для большинства случаев погрешность.

Окно установки параметров (свойств) идеального интегратора, показанное на рис. 3.27 снизу, позволяет задать следующие параметры:

- Initial Condition — начальное значение y(tstart);
- ID — идентификационный номер интегратора;
- Checkpoint State — зафиксированное состояние интегратора;
- Label — задание метки блока.

Полезно сделать некоторые замечания о параметрах интегратора. Начальное значение Initial Condition по умолчанию задано нулевым. Однако его можно сделать заданным исходя из условий физической реализации задачи. Некоторые параметры, например напряжение на конденсаторе или ток в индуктивности, не могут изменяться скачком, и при интегрировании нужно задавать соответствующие начальные условия.

Многие системы и устройства для моделирования требуют применения ряда интеграторов, число которых задает порядок модели. Параметр ID задает идентификационный номер модели интегратора.

При работе интеграторы используют эффект памяти. Опция Checkpoint State на вкладке Preferences окна свойств моделирования позволяет при остановке моделирования записать состояние интеграторов в файл. В этом случае

для заданного интегратора его состояние будет отражено значением параметра Checkpoint State.

3.6.2. Блок интегрирования с насыщением limitedIntergator

Блок интегрирования с насыщением возвращает сигнал, который может изменяться только в том случае, если он находится в заданных пределах $l < y < u$. Если y достигает значения нижнего порога l (от слова low — нижний), то дальнейшее уменьшение сигнала на выходе ограничивается значением l. Аналогично, если сигнал на выходе достигает верхнего предела u (от слова upper — верхний), он перестает изменяться.

Подобный эффект можно создать, установив на выходе идеального интегратора ограничитель выходного сигнала по уровню. Однако это не совсем верное решение. Блок limitedIntergator более корректно отслеживает работу функции интегрирования и рекомендуется к применению в том случае, когда надо ограничивать пределы изменения выходного сигнала. Блок интегратора с насыщением имеет три входа и один выход. На верхний вход подается интегрируемый сигнал, а на входы ниже — пороговые сигналы l и u. Применение блока показано на рис. 3.28. Окно установки параметров этого блока аналогично представленному на рис. 3.28.

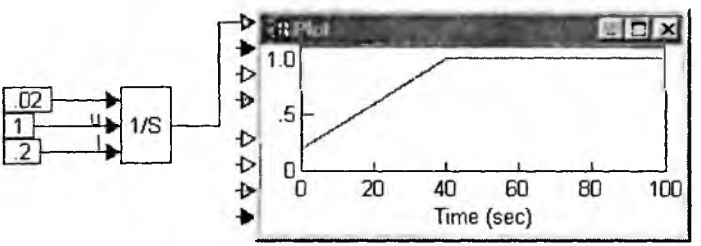


Рис. 3.28. Пример интегрирования с насыщением

Этот блок можно уподобить интегратору на операционном усилителе, у которого уровни выходного напряжения ограничены некоторыми порогами. Подобные интеграторы часто применяются и при моделировании других устройств.

3.6.3. Блок интегрирования со сбросом resetIntegrator

Еще один тип интегратора в системе VisSim представлен блоком интегрирования со сбросом resetIntegrator. Этот блок также имеет три входа и один выход. Верхний вход служит для подачи подлежащего интегрированию сигнала, средний — для подачи управляющего сигнала b и нижний — для задания уровня сброса r. Если уровень сигнала на управляющем входе по абсолютному значению меньше 1, то интегратор интегрирует как обычно. В противном случае интегратор сбрасывается и прекращает интегрирование. При этом сигнал на его выходе устанавливается равным значению сигнала на нижнем входе r.

Наглядный пример работы интегратора со сбросом дан на рис. 3.29. Здесь сигнал сброса формируется с помощью двух ступенек и блока суммирования

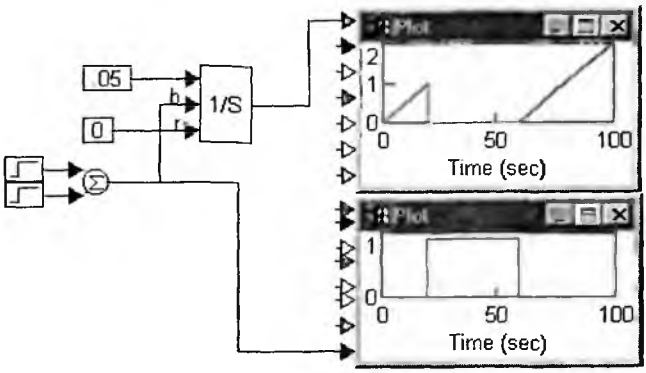


Рис. 3.29. Пример применения интегратора со сбросом

Одна ступенька (единичный положительный перепад) формируется с задержкой на 20 с, а другая (единичный отрицательный перепад) с задержкой в 60 с. В результате на выходе схемы суммирования создается прямоугольный импульс с амплитудой 1, задержкой 20 с и длительностью 40 с. Он и используется для сброса интегратора.

Окно установки параметров у интегратора со сбросом такое же, как у идеального интегратора (см. рис. 3.28).

3.7. Блоки анализа линейных систем

3.7.1. Блок анализа по уравнениям состояния stateSpace

Как известно, линейные системы могут быть описаны в стандартной векторной форме пространства состояния системы (см. главу 1):

$$u' = \mathbf{A}u + \mathbf{B}x$$

$$y = \mathbf{C}u + \mathbf{D}x,$$

где x — вектор входных переменных; y — вектор выходных переменных; u — вектор переменных состояния (фазовых координат системы); **A** — матрица коэффициентов системы; **B** — матрица входных коэффициентов (матрица управления); **C** — матрица выходных коэффициентов; **D** — матрица коэффициентов пропорциональных каналов (матрица компенсации). Максимальная размерность пространства состояний, которую поддерживает VisSim, — 90 фазовых координат.

Блок stateSpace (пространство состояний) предназначен для моделирования системы в пространстве состояний. Он может иметь много входов и много выходов. Матрицы коэффициентов пространства состояний определяются в подключаемых к блоку .m или .mat-файлах, которые могут быть созданы следующими способами.

1. С помощью библиотеки VisSim/Analyze, которая имеет функцию линеаризации фрагмента блок-схемы с одним входом и одним выходом. При этом можно получить матрицы **A**, **B**, **C** и **D**, которые определяются в пр

цессе моделирования. Они могут быть либо выведены на экран, либо сохранены в .m-файле.

2. С помощью файла с расширением .m (m-файла) может быть создан текстовым редактором, например редактором Блокнот или редактором m-файлов системы MATLAB.

Пример применения этого блока описан ниже.

3.7.2. Примеры с применением VisSim и системы MATLAB

Наиболее мощными средствами по работе с пространствами состояний обладает матричная система MATLAB. При подготовке m-файла в среде MATLAB надо придерживаться следующих правил:

- каждую новую матрицу надо описывать с новой строки;
- содержимое матриц заключается в квадратные скобки, после закрывающей скобки ставится точка с запятой;
- элементы матриц отделяются пробелами;
- строки матриц отделяются точкой с запятой;
- если матрица большая, то ее строки можно начинать с новой строки;
- в файле не следует использовать команды MATLAB.

Особенности применения блока stateSpace рассмотрим на примере, представленном на рис. 3.30. Здесь вычисляется реакция непрерывной линейной системы второго порядка на скачок в момент t = 0.

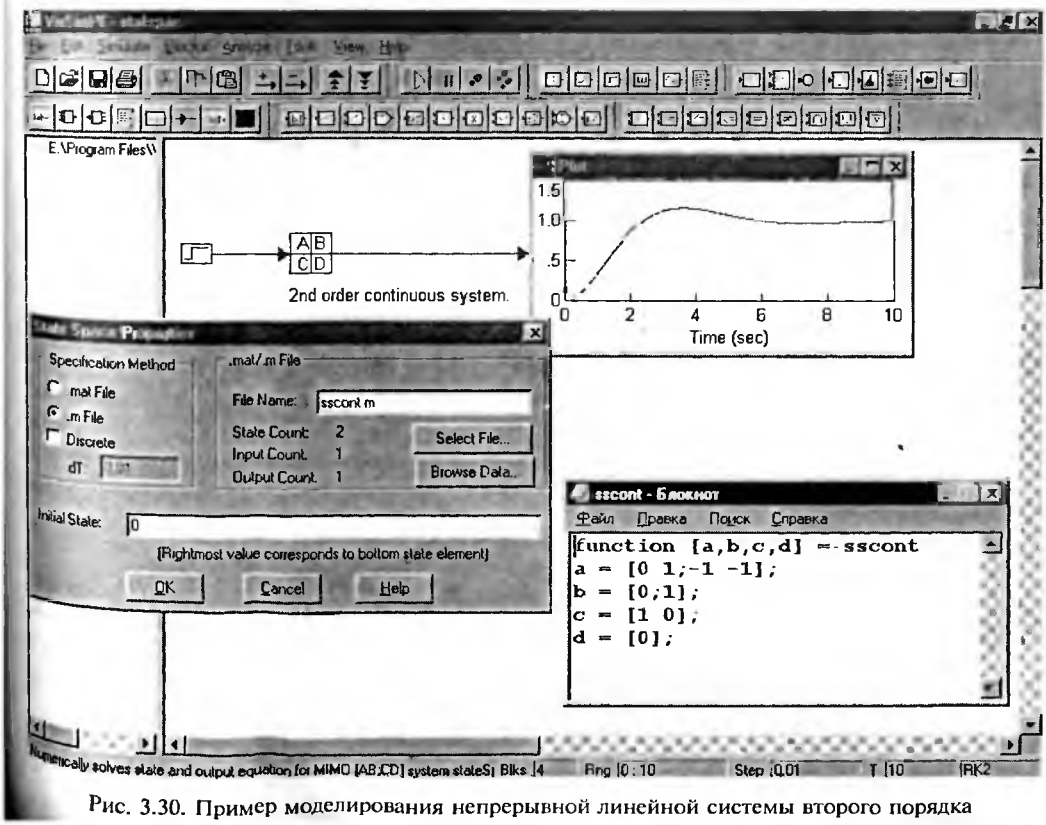


Рис. 3.30. Пример моделирования непрерывной линейной системы второго порядка

Окно свойств блока представлено снизу и слева. В левой его части указывается тип файла с данными пространства состояния системы. Отсутствие птички у опции Discrete означает, что анализируется непрерывная система. В правой части окна показывается имя файла и указывается число переменных, входов и выходов. Кнопка Select File... открывает стандартное окно Windows для загрузки файла. А кнопка Browse Data... открывает окно текстового редактора Блокнот с текстом текущего (загруженного) файла, составленного по описанным выше правилам. Для моделируемой системы это окно показано снизу и справа в окне модели.

В окне свойств имеется также возможность задания через пробелы начальных условий (значений фазовых координат u), которые будут установлены в матрице интеграторов [1/S] на первом шаге моделирования. Самое правое значение будет установлено на нижнем интеграторе в матрице. Неопределенные начальные значения принимаются равными 0.

Пример моделирования дискретной линейной системы второго порядка по заданному пространству ее состояний представлен на рис. 3.31. Для перехода к анализу дисретных систем в окне свойств блока stateSpace надо выставить птичку в опции Diskrete и задать шаг во времени dT.

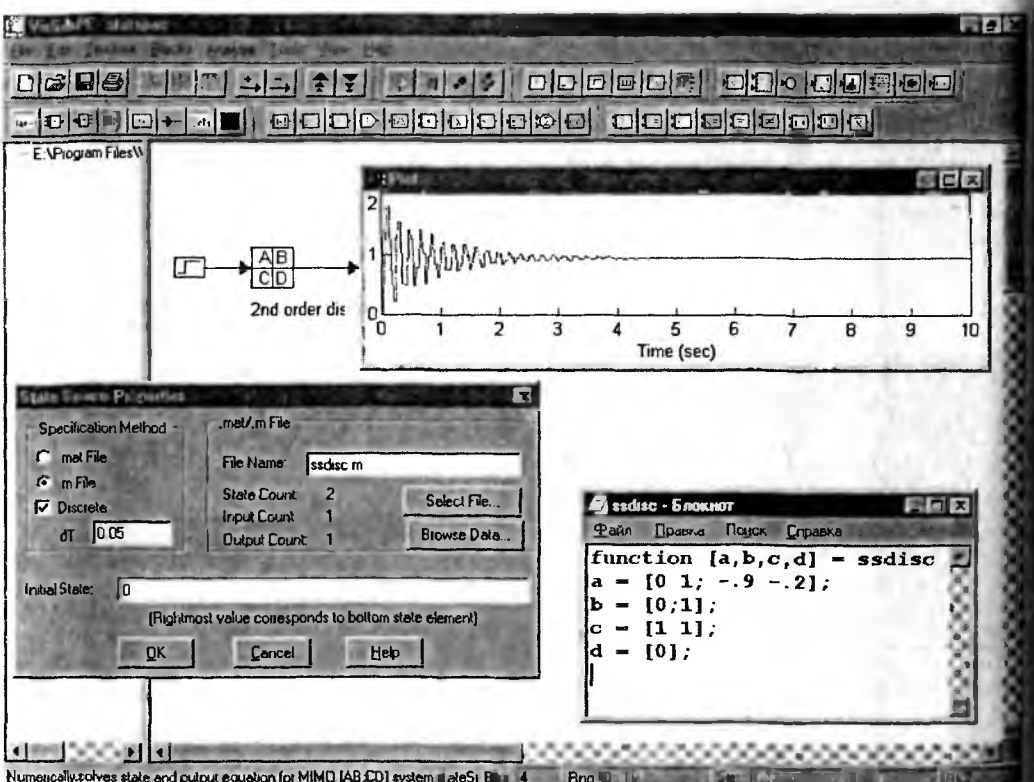


Рис. 3.31. Пример моделирования дискретной линейной системы второго порядка

3.7.3. Блок передаточной функции transferFunction

Блок transferFunction преобразует входной сигнал x в любой момент времени в выходной сигнал y в соответствии с операторной передаточной функцией, которая может быть определена отношением полиномов:

$$\frac{y}{x} = \frac{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0}{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}.$$

Здесь передаточная функция определена так, как это дано в документации по системе VisSim. Стоит напомнить, что в нашей литературе (см. главу 1) принято определять y как $y(s)$, а x как $x(s)$. Кроме того, коэффициенты знаменателя обозначаются как a, а числителя как b. Возможны следующие пути задания передаточной функции:

- заданием коэффициентов полиномов числителя и знаменателя;
- заданием корней полиномов числителя и знаменателя (нулей и полюсов);
- загрузкой из .m или .mat-файла ABCD матриц коэффициентов пространства состояний;
- применением коэффициентов БИХ-фильтра (с бесконечной импульсной характеристикой);
- применением коэффициентов КИХ-фильтра (с конечной импульсной характеристикой).

Рисунок 3.32 иллюстрирует расчет реакции на скачок двух линейных систем, заданных своими передаточными характеристиками. Одна из систем непрерывная, другая — дискретная.

На рис. 3.33 показано окно свойств блока transferFunction для случая моделирования линейной дискретной системы — пример, представленный на рис. 3.32 снизу. Спецификация передаточной функции в этом окне позволяет задавать функцию в виде отношения полиномов или из файлов. В последнем

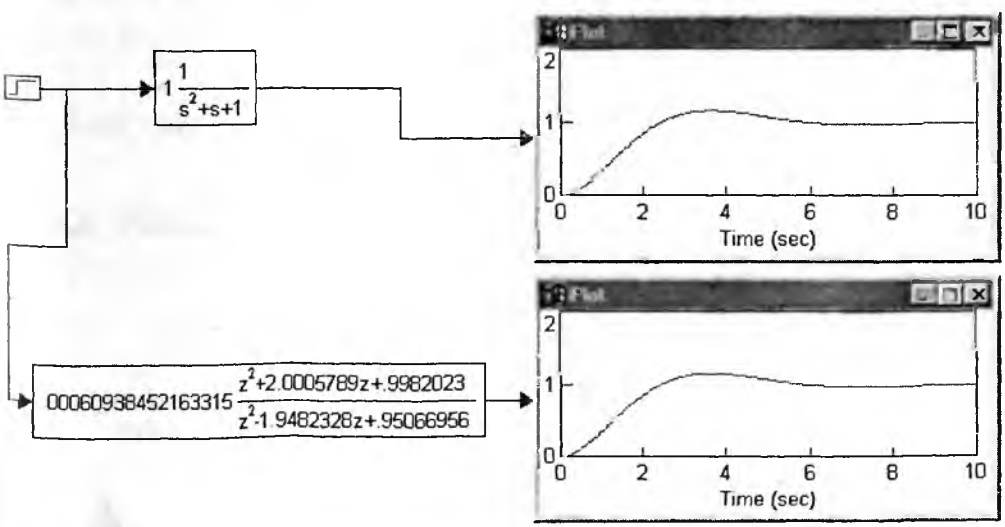


Рис. 3.32. Реакция линейных систем, заданных передаточными характеристиками, на скачок

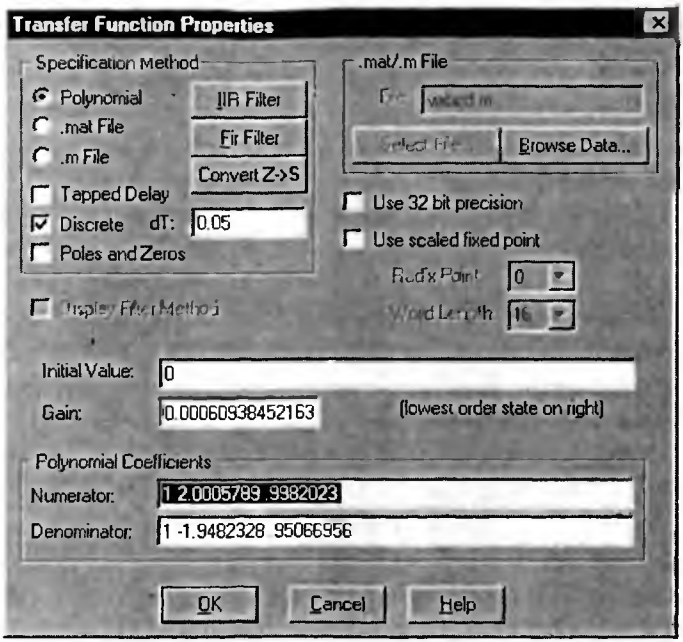


Рис. 3.33. Окно свойств блока transferFunction при моделировании линейной системы

случае можно просмотреть запись файла аналогично тому, как просматрив ются файлы пространства состояний. В случае задания передаточной фун ции в виде отношения полиномов нужно задать параметр Gain и списки ко эффициентов числителя (Numerator) и знаменателя (Denominator).

В окне свойств системы можно также задать моделирование непрерывны систем. Для моделирования дискретной системы надо задать интервал врем ни дискретизации *dT*. Кнопка Convert Z->S (или S->Z) позволяет задавать п редаточную функцию в операторном виде или в виде, характерном для Z-пр образований.

3.8. Матричные операции раздела библиотеки Matrix Operations

3.8.1. Блок буфера buffer

Блок буфера buffer накапливает последовательность входных значений создает на выходе шину (матрицу-строку) данных. Буфер реализует принц «первый вошел — первый вышел» (FIFO). Это означает фактическую разве ку сигнала с конца его последовательности в начало. Нередко это приводи путанице в трактовке получаемых после вывода из буфера временных д грамм и даже к ошибкам в их интерпретации. Поэтому рассмотрим работу б фера более подробно.

Как видно из рис. 3.34, буфер имеет два настраиваемых параметра — чи ло ячеек и шаг во времени *dT*. В примерах этого рисунка ячеек буфера 100

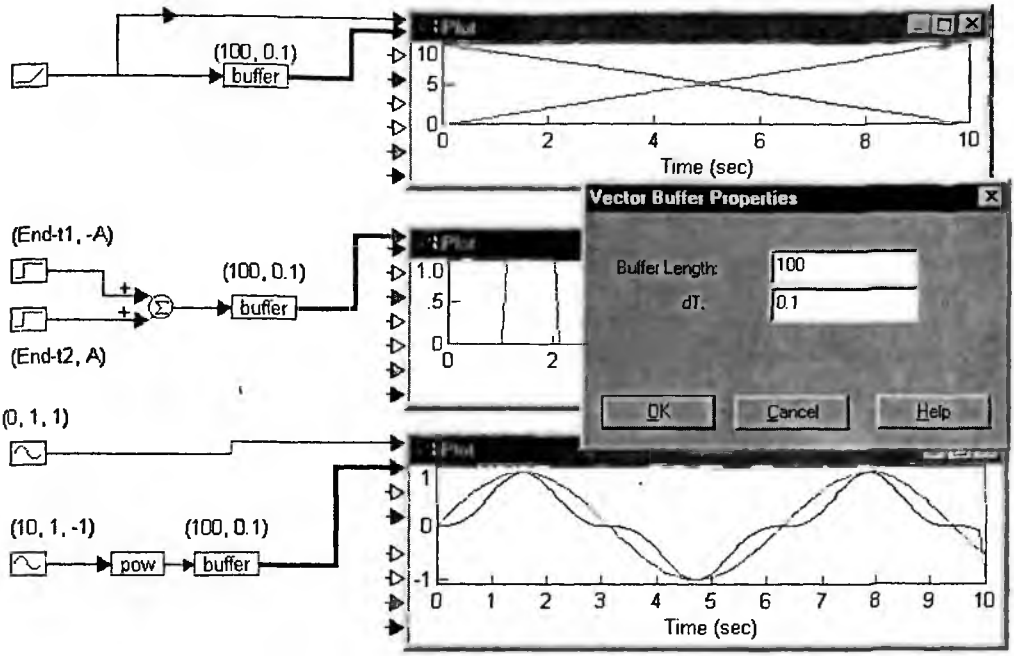


Рис. 3.34. Примеры работы блока buffer

число шагов моделирования (начальное время Start = 0, конечной End = 10 и шаг Step Size = 0,1). В примере, показанном сверху, в буфер подается нарастающий сигнал с 0 до 10 (с параметром slope = 1). На выходе буфера этот сигнал растет с 0 до 10 от конца осциллограммы к ее началу.

Во втором примере показано, как сформировать на выходе буфера прямоугольный импульс с заданным положением переднего t1 и заднего t2 фронтов. Для этого надо сформировать отрицательный перепад с амплитудой –A (по умолчанию A = 1) с задержкой End-t1, а затем сформировать положительный перепад с амплитудой A и задержкой End-t2.

Чтобы сформировать синусоидальный (или синхронный с ним сигнал) нужно сменить его полярность и ввести задержку, равную времени End. Это и показывает нижний пример на рис. 3.34. Здесь на выходе буфера формируется сигнал в виде синуса в кубе. Буфер часто используется при проведении спектрального анализа, и в этом случае знание особенностей его работы крайне необходимо.

3.8.2. Блок точечного произведения dotProduct

Блок dotProduct вычисляет сумму всех произведений соответствующих элементов в двух матрицах-столбцах или строках $x1$ и $x2$ одинакового размера n:

$$y = \sum_{k=1}^{n} x1_k \bullet x2_k .$$

Пример применения блока dotProduct для вычисления точечного (скалярного) произведения двух векторов с размером n = 4 показан на рис. 3.35.

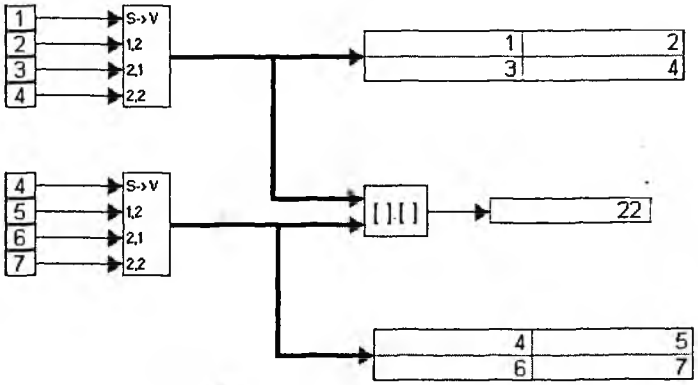


Рис. 3.35. Пример применения блока dotProduct

3.8.3. Блоки прямого fft и обратного ifft быстрых преобразований Фурье

Блок fft выполняет операцию быстрого преобразования Фурье (БПФ), в результате которой временная зависимость сигнала, представленного выборками, преобразуется в спектр гармоник — гармонических сигналов с частотами кратными частоте повторения сигнала. Описание преобразований Фурье было дано в главе 1, а описание БПФ можно найти в учебнике [39]. Последнее резко повышает скорость преобразования, особенно если сигнал можно представлять 2*n* выборками (4, 8, 16, 32, 64, 128, ...). Если на вход блока подается количество выборок не кратное степеням двойки, то блок автоматически дополняет массив выборок нулями. Размер выходного массива совпадает с размером входного массива — 2n. Возвращаемые данные — это пары коэффициентов ряда Фурье пропорциональные вещественным и мнимым составляющим каждой гармоники.

Блок ifft выполняет операцию обратного БПФ, в результате которой данные частотный спектр гармоник преобразуется во временной сигнал. После прямого преобразования (спектрального анализа) и затем обратного (спектральный синтез) происходит восстановление (реставрация) отсчетов исходного сигнала.

Рисунок 3.36 показывает спектральный анализ, а затем синтез исходного сигнала в виде синусоидальной функции, возведенной в кубическую степень, прошедшего через буфер. Построены графики исходного сигнала, коэффициентов Фурье, вычисленных при спектральном анализе, и результаты синтеза (реставрации) сигнала.

Еще один пример прямого и обратного преобразований Фурье для прямоугольного импульса — меандр, полученный из преобразования синусоиды функцией sign, показан на рис. 3.37. Здесь отчетливо видна особая роль буфера. Благодаря ему преобразования Фурье выполняются только для тех частей сигнала, которые следуют за перепадом. Такое преобразование называют оконным преобразованием Фурье, причем окно имеет прямоугольную форму.

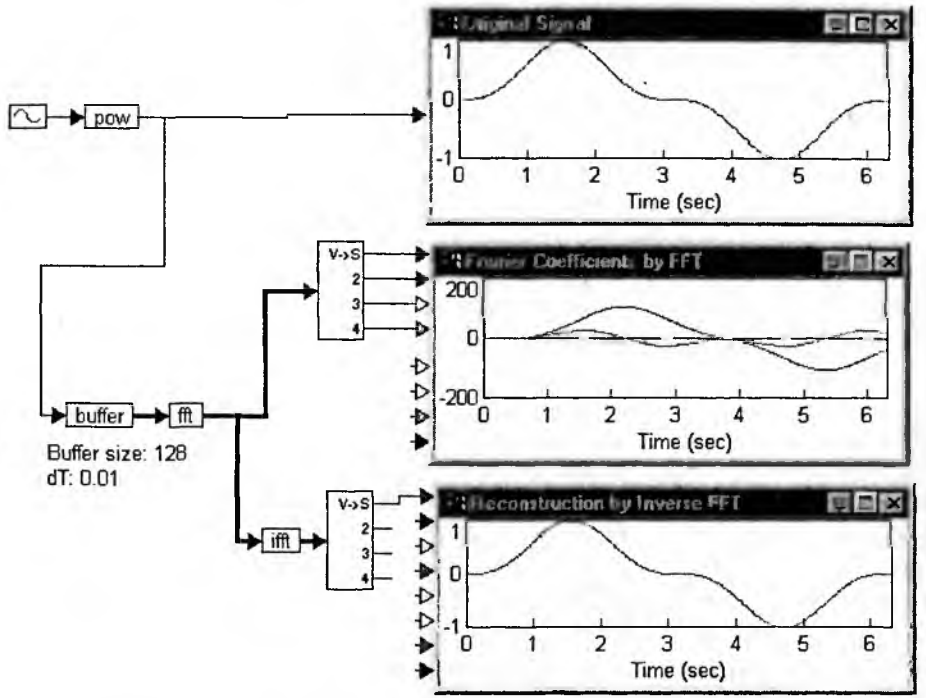


Рис. 3.36. Прямое и обратное быстрые преобразования Фурье для сигнала в виде синусоиды, возведенной в третью степень

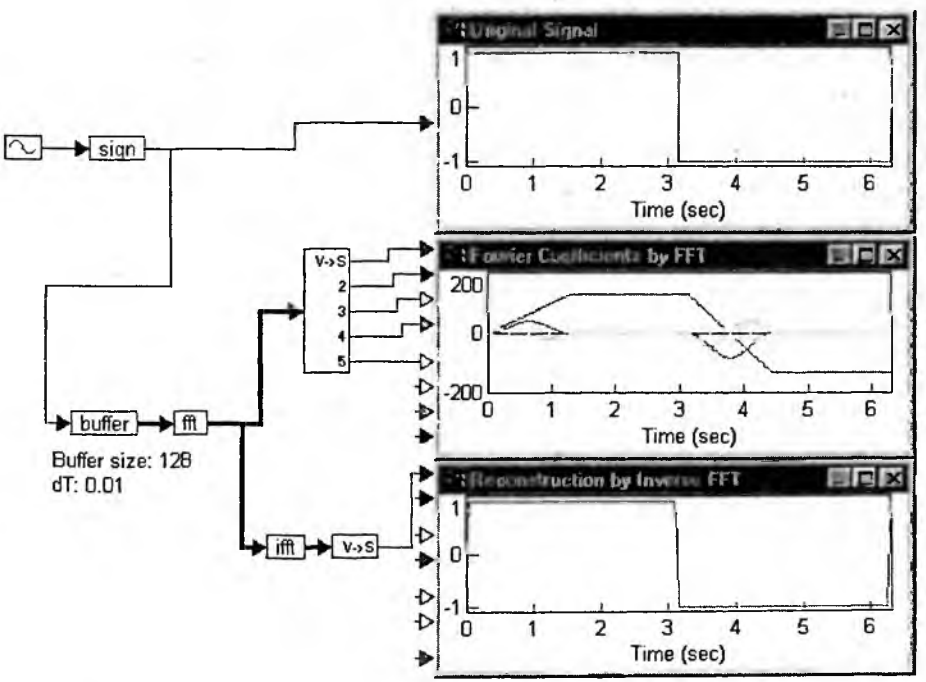


Рис. 3.37. Прямое и обратное быстрые преобразования Фурье для сигнала в виде меандра

В описанных примерах следует помнить, что преобразования Фурье выполняются для отсчетов сигналов, прошедших через буфер. Если при обратном преобразовании Фурье использовать только часть гармоник, полученных при прямом преобразовании, то погрешность восстановления заметно возрастает и начинает сказываться эффект Гиббса.

Для повышения разрешения при спектральном анализе может использоваться оконное преобразование Фурье, достаточно подробно описанное в [37], как и альтернативное ему вейвлет-преобразование. Для осуществления оконного преобразования Фурье специальных блоков в VisSim нет. По существу, прямоугольное окно создает блок buffer, а любое другое окно легко создать, умножив сигнал на функцию окна с помощью блока умножения.

3.8.4. Блок доступа по индексам index

Блок доступа по индексам обеспечивает доступ к отдельному элементу вектора или матрицы путем указания двух его индексов — номера строки *r* и столбца *c* (отсчет с 1). Если будет предпринята попытка адресации элемента вне границ матрицы, то VisSim отобразит сообщение о выходе первого или второго индекса за размерность матрицы и установит флаг ошибки, закрасив блок элемент красным цветом.

При индексировании сигналами *c* и *r* действуют правила:

- если индексирующий сигнал имеет тип с плавающей точкой, то округление для индексации выполняется отбрасыванием дробной части. Например, сигнал, имеющий значение 1.98, будет округлен до 1 (т. е. будет индексировать первый, а не второй элемент);
- если не подключена входная шина массива, то при любых значениях индексирующих сигналов блок элемент будет возвращать 0 без предупреждений об ошибке;
- при адресации элемента вне границ входной матрицы и игнорировании предупреждающих сообщений блок-элемент будет возвращать число очень близкое к нулю, например 1.06983e-306.

Пример применения блока index приведен на рис. 3.38.

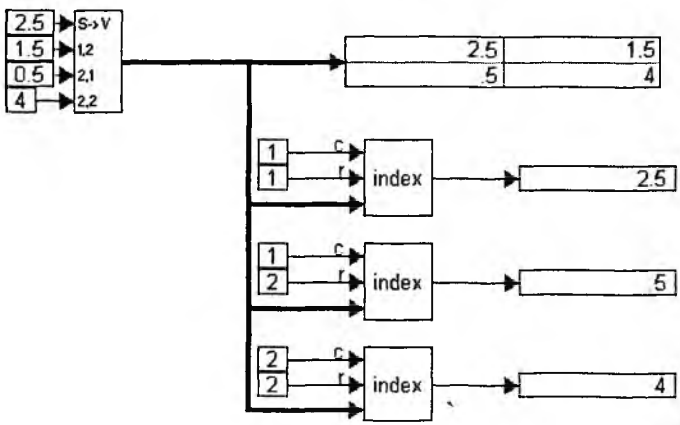


Рис. 3.38. Применение блока index для доступа к элементам матрицы

Этот пример настолько прост, что не нуждается в особых комментариях. Окно параметров блока index позволяет задавать только метку для блока.

3.8.5. Блок инвертирования матрицы invert

Блок invert возвращает квадратную матрицу обратную входной квадратной матрице, которая должна быть невырожденной (неособенной). Обратная матрица есть частное от деления единичной матрицы на исходную матрицу. Ее размер равен размеру входной матрицы.

Наиболее часто инвертирование (обращение) матрицы используется для решения систем линейных уравнений, записанных в матричном виде $\mathbf{A}\times\mathbf{X} = \mathbf{B}$, где **A** — матрица коэффициентов правой части системы уравнений, **X** — вектор неизвестных и **B** — вектор свободных членов (правая часть системы). Из этой записи вытекает, что $\mathbf{X} = \mathbf{A}^{-1}\times\mathbf{B}$. Пример решения системы из двух линейных уравнений с применением этого метода решения представлен на рис. 3.39.

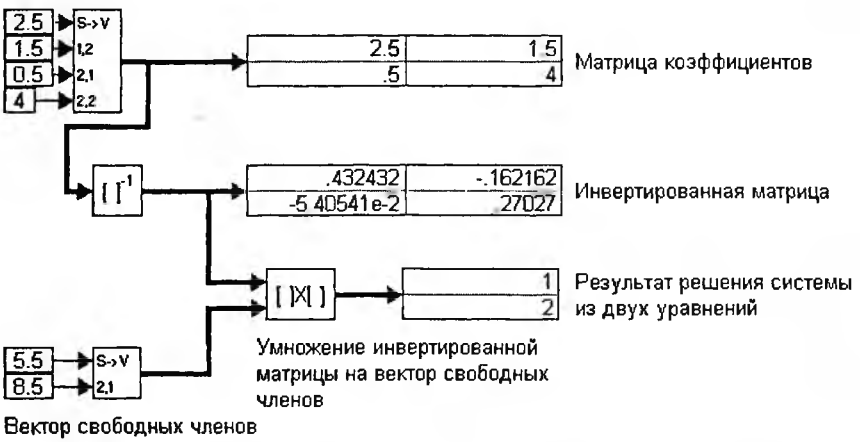


Рис. 3.39. Пример решения системы из двух линейных уравнений с применением инвертирования матрицы коэффициентов системы

Если требуется выполнить операцию поэлементного масштабирования сигналов в матрице-шине, то следует пользоваться блоком / (деление). А если требуется выполнить поэлементную операцию нахождения обратных величин для сигналов в матрице-шине, то следует пользоваться блоком 1/X. Не следует путать эти операции с операцией инвертирования матрицы.

3.8.6. Блок перемножения массивов multiply

Для перемножения двух массивов необходимо использовать блок multiply, корректно выполняющий эту операцию. Этот блок имеет обозначение в виде прямоугольника с надписью [ ]×[ ]. Он имеет два шинных входа и один выход. Пример применения блока для перемножения инвертированной матрицы на вектор свободных членов при решении системы линейных уравнений в матричной форме дан на рис. 3.39.

Операция перемножения может выполняться для «сцепленных» матриц, имеющих равное число строк, или для квадратных матриц. Возможно также умножение матрицы на вектор. Эту операцию не стоит путать с операцией почленного умножения двух массивов. Последняя реализуется блоком умножения *.

3.8.7. Блок транспонирования массивов transpose

Блок транспонирования массивов transpose служит для выполнения операции транспонирования Y = XT или [Xij]T = [Xji]. Для матрицы эта операция означает перемену местами столбцов и строк. Повторное применение операции восстанавливает исходную матрицу (см. пример на рис. 3.40).

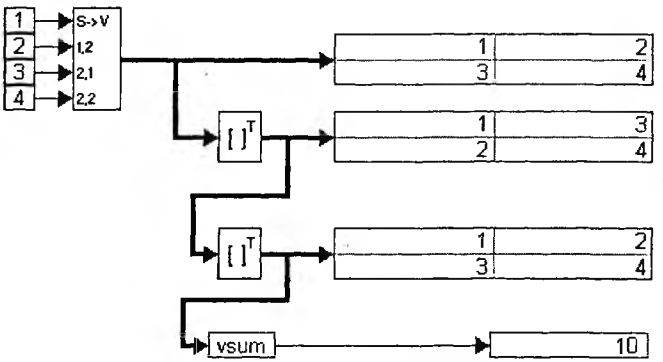


Рис. 3.40. Примеры выполнения операций транспонирования матрицы и суммирования ее элементов

Для вектора транспонирование означает смену типа, например вектор-строка превращается в вектор-столбец.

3.8.8. Блок суммирования элементов массивов vsum

Блок суммирования элементов массивов vsum служит для вычисления суммы всех элементов массива, т. е. величины

$$y = \sum x_{i,j}.$$

Пример применения этого блока дан на рис. 3.40 снизу.

3.8.9. Блок вычисления спектра мощности psd

Блок psd (впервые появился в VisSim 4.5) возвращает спектр мощности сигнала, используя быстрое преобразование Фурье для массива выборок и соответствующее масштабирование. При увеличении амплитуды входного сигнала в 10 раз спектр поднимается на 40 дБ, поскольку мощность увеличивается пропорционально квадрату сигнал. Представление спектра в виде спектра мощности зачастую дает более детальную спектральную характеристику, чем только при использовании блока fft.

Если на вход блока подается количество выборок (n) не кратное степеням двойки (8, 16, 32, 64, 128, ...), то блок автоматически дополняет массив выборок нулями. Размер выходного массива в два раза меньше размера входного массива. Шаг частоты между спектральными полосами (для выходных данных блока psd) определяется по формуле: (1/2 частота_стробирования_буфера) / (n/2). Доступ к индивидуальным полосам спектра возможен с помощью блока VecToScalar.

Рисунок 3.41 показывает типичный пример спектрального анализа для прямоугольного импульса единичной амплитуды и длительности $t_и = 0.1$ с. Импульс создается двумя ступеньками, вторая из которых сдвинута на 0,1 с и имеет высоту –1. Буфер (120 разрядов шины при $dT = 0.01$) превращает последовательность отсчетов сигнала во времени в параллельный код и сдвигает импульс (сдвиг для спектрального анализа значения не имеет).

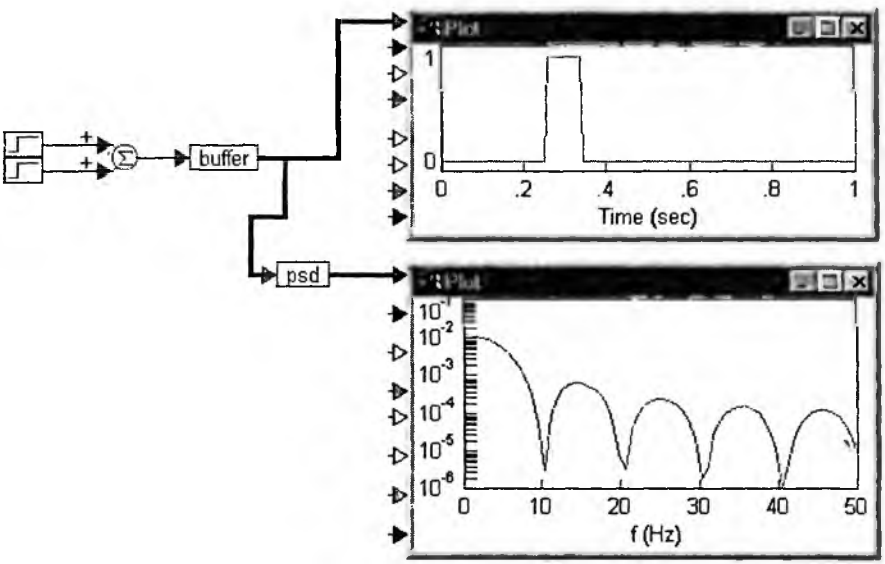


Рис. 3.41. Спектральный анализ прямоугольного импульса с помощью блока psd

Для представления спектра (нижний осциллограф) надо соответствующим образом отформатировать график, например задать логарифмический масштаб для уровня спектральной плотности по мощности и уточнить диапазон частот. В нашем случае нули (провалы) спектра должны быть на частотах $k \cdot t_и$, где $k = 1, 2, \ldots$. Следует также сменить обозначение по горизонтальной оси с времени на частоту.

3.9. Нелинейные блоки

3.9.1. Блок мультиплексора case

Блок мультиплексора case подключает выход к одному из входов, обозначенных как 0, 1, 2 и т. д. По умолчанию сигнальных входов три, но их число можно увеличить. Мультиплексор имеет управляющий вход (верхний), сигнал case на котором и определяет тот вход, к которому подключается выход.

При адресации сигнальных входов в отношении сигнала case действуют следующие правила:

- Адресный диапазон начинается с нуля. Нулевой адрес указывает на вход 0 (второй с верху). Адрес 1 указывает на вход 1 и т. д.
- Если сигнал case имеет тип с плавающей точкой, то округление для адресации выполняется отбрасыванием дробной части. Например, сигнал, имеющий значение 0.99, будет округлен до 0, т. е. будет подключен нулевой, а не первый канал.
- Если сигнал case выйдет за адресный диапазон мультиплексора, то VisSim отобразит сообщение об этом и установит флаг ошибки, закрасив блок красным цветом.

Примеры работы блока case представлены на рис. 3.42. В комментариях они не нуждаются.

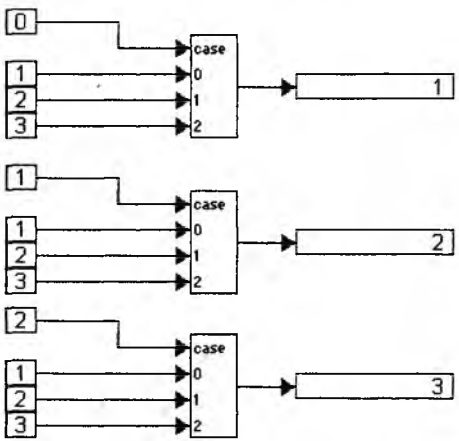


Рис. 3.42. Примеры применения блока case

3.9.2. Блок детектора переходов crossDetect

Блок детектора переходов имеет один вход и один выход. В окне свойств этого блока помимо установки метки задается уровень перехода Cross Point, по умолчанию равный 0. Блок фиксирует переход сигнала на входе через заданный уровень. При этом он вырабатывает на выходе следующий сигнал:

- y = –1, если входной сигнал x пересек уровень сверху;
- y = 1, если входной сигнал x пересек уровень снизу;
- y = 0, иначе.

Блок генерирует импульс единичной амплитуды, если входной сигнал на текущем шаге моделирования пересекает заданный уровень. При этом знак импульса соответствует знаку приращения сигнала. Если пересечение не фиксируется, то на выходе блока будет нулевое значение. Факт пересечения обнаруживается благодаря внутреннему буферному регистру, который запоминает входной сигнал на один шаг моделирования.

Рисунок 3.43 демонстрирует работу детектора переходов через нуль. Здесь показано превращение синусоиды в короткие импульсы с удвоенной частотой повторения.

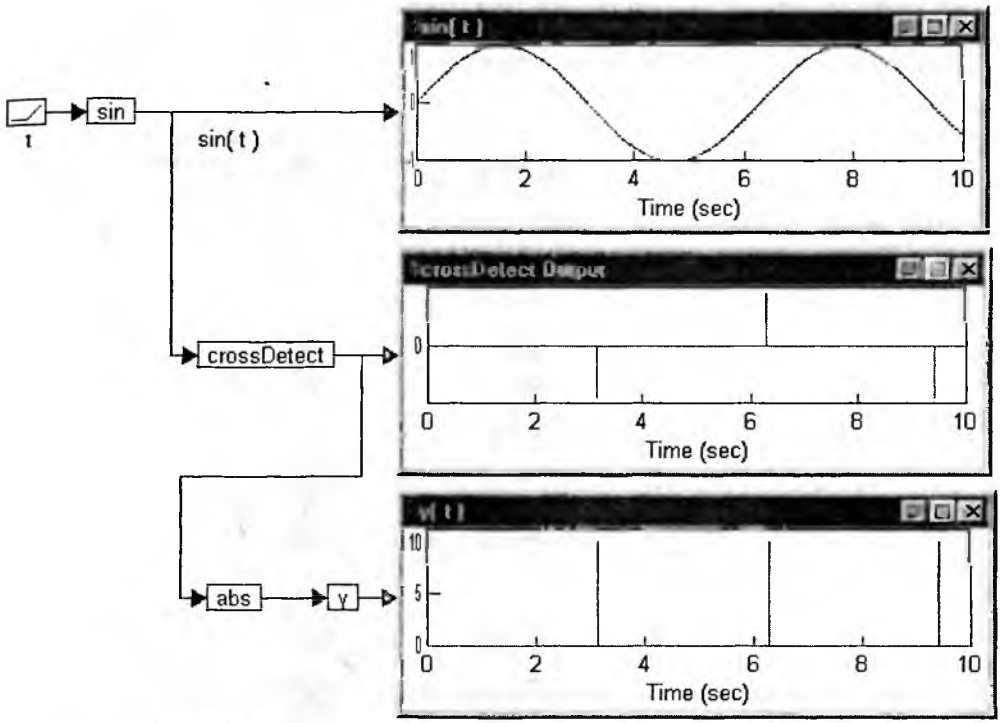


Рис. 3.43. Применение блока детектора перехода сигнала через нуль

3.9.3. Блок с «мертвой» зоной deadBand

Блок с «мертвой» зоной (или зоной нечувствительности) deadBand возвращает сигнал на выходе *y*, пропорциональный сигналу на входе *x*, за исключением случая изменения *x* в пределах некоторой мертвой зоны deadBand. Передаточная характеристика этого блока, т. е. зависимость *y*(*x*), описывается следующими выражениями:

y = 0, если | x | ≤ (deadBand/2)

y = x – sign x * (deadBand/2), иначе.

Пример применения блока представлен на рис. 3.44.

Данный вид нелинейности используется в механических системах для имитации люфта. В электронных усилителях данный вид нелинейности имитирует искажения типа ступеньки. Окно свойств блока позволяет задать величину зоны нечувствительности (по умолчанию 0.2) и определить метку для блока.

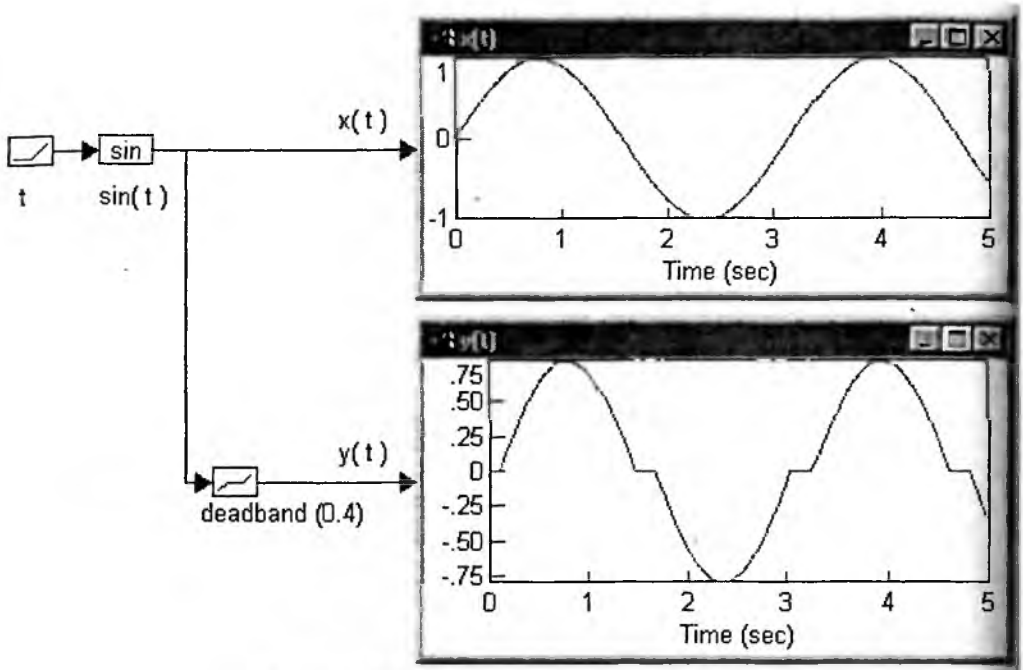


Рис. 3.44. Пример применения блока deadBand

3.9.4. Блок отбрасывания дробной части int

Блок int превращает число *x* с дробной частью в целое число просто отбрасыванием дробной части (рис. 3.45). Например, сигнал, имеющий значение 4.999, будет округлен до 4. Блок обрабатывает только отдельные сигналы, и его вход нельзя подключать к шине со многими значениями сигнала. Окно свойств блока позволяет задавать только его метку.

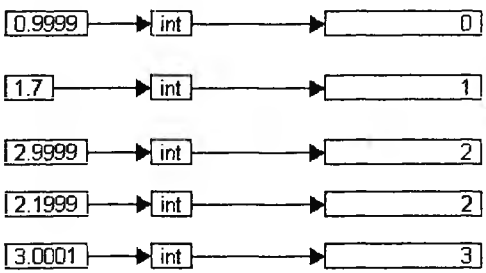


Рис. 3.45. Работа блока int

3.9.5. Блок двухстороннего ограничения limit

Блок двухстороннего ограничения limit имитирует работу двухстороннего ограничителя со следующей передаточной характеристикой:

y = x, если lb ≤ x ≤ ub
y = lb, если x < lb
y = ub, если x > ub

Блок ограничивает выходной сигнал в соответствии с заданными пределами — верхним ub, и нижним lb. Если входной сигнал находится в заданных пределах, то блок предает его на выход без преобразований. Если входной сигнал выходит за пределы, то до тех пор, пока ситуация не изменится, выходному сигналу будет присваиваться значение соответствующего предела. Верхний ub (Upper Bound) и нижний lb (Lower Bound) пределы задаются в диалоговом окне свойств блока. По умолчанию используются значения 100 и -100. Возможность их динамического изменения отсутствует. Блок обрабатывает только отдельные сигналы (нельзя подключить шинный проводник к входу). Применение блока и его окно свойств представлены на рис. 3.46.

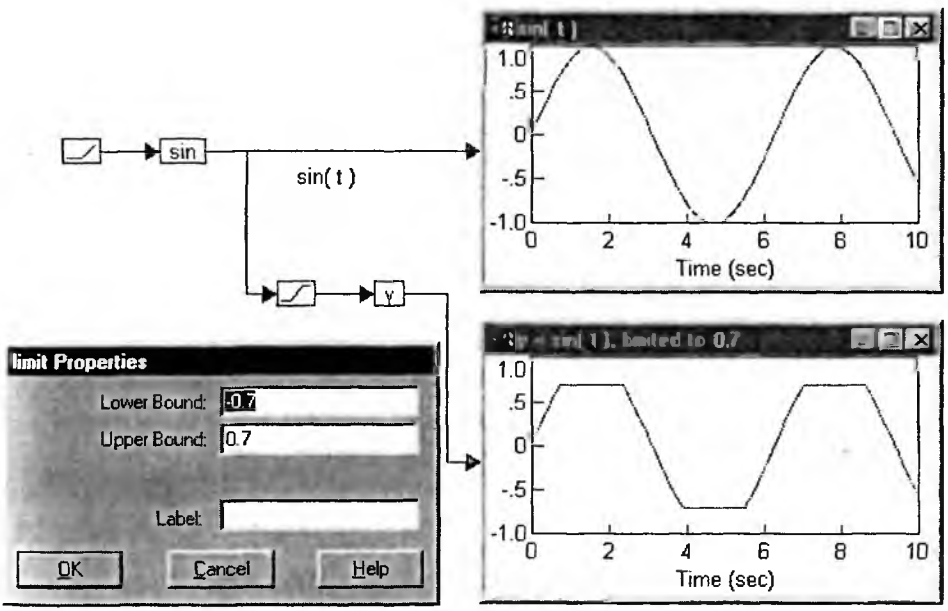


Рис. 3.46. Применение блока limit и окно его свойств

Блок имитирует различные ограничения, например сжатия и растяжения пружины или обрезания верхушек сигнала в электронном усилителе.

3.9.6. Блок таблицы map

Блок таблицы map преобразует входной сигнал в соответствии с произвольно определяемой нелинейной функцией, которая в динамике может зависеть от одного или двух параметров. Работа блока задается выражениями:

y = *.map [x1] или y = *.map [x1, x2] или y = *.map [x1, x2, x3],
где *.map — файл с 1D, 2D или 3D-матрицей

Окно свойств блока показано на рис. 3.47 с примером загрузки файла 3d_ex.map. В окне отображается содержание выбранного файла. Можно вывести окно загрузки и окно просмотра файла, активизируя кнопки Select File... и Browse Data... соответственно. Можно также задать тип данных и их интерполяцию и экстраполяцию.

Пример применения блока map представлен на рис. 3.48. В данном случае блок используется для тройного повторения сигнала.

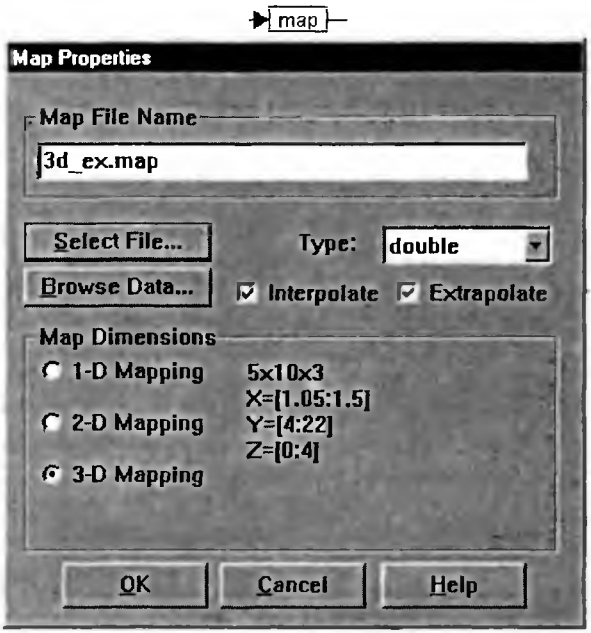


Рис. 3.47. Окно свойств блока map

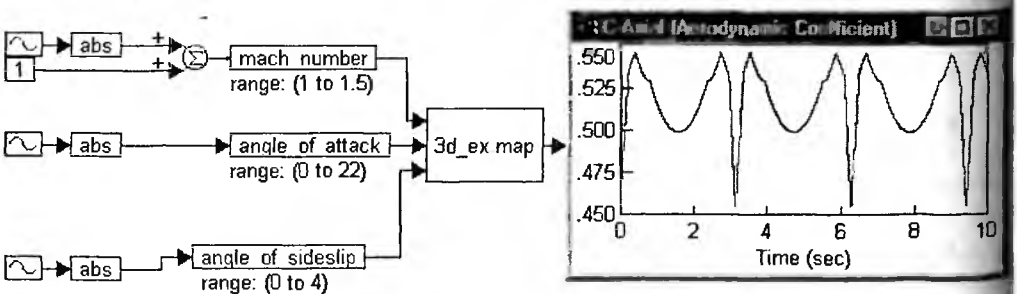


Рис. 3.48. Пример применения блока map

3.9.7. Блок выделения максимального значения max

Блок max сравнивает значения двух входных сигналов на каждом шаге симуляции и больший из них передает на выход. Его работа описывается выражениями:

$y = x1$, если $x1 > x2$ $\qquad$ $y = x2$, если $x1 < x2$

Рисунок 3.49 поясняет работу блока max. На вход блока подаются два сигнала — синусоиды и шума, создаваемого генератором случайных чисел. Время от времени тот или иной сигнал оказывается максимальным. Временные зависимости сигналов на входах блока max и на его выходе дают прекрасное представление о характере работа блока.

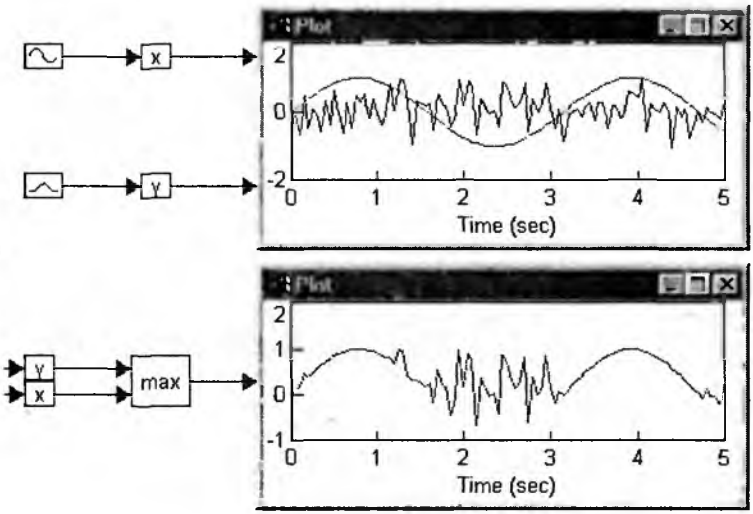


Рис. 3.49. Работа блока max

Окно свойств блока max никаких настроек не содержит. Можно лишь задать метку блока.

3.9.8. Блок merge

Блок merge осуществляет условную коммутацию двух сигнальных проводников или шин (xt и xf) в зависимости от значения логического управляющего сигнала xb. Его работа описывается выражениями:

y = xt , если | xb | ≥ 1
y = xf , если | xb | < 1

Обозначения b, t, и f, которыми помечены входы блока, — это первые буквы слов boolean, true и false. Блок фактически реализует конструкцию условного исполнения if ... then ... else ..., что поясняется примером, представленным на рис. 3.50.

3.9.9. Блок выделения минимума min

Блок min сравнивает значения двух входных сигналов на каждом шаге симуляции и меньший из них передает на выход. Это соответствует следующим выражениям:

y = x1, если x1 < x2
y = x2, если x1 > x2

Рисунок 3.51 поясняет работу блока max. На вход блока подаются два сигнала — синусоиды и шума, создаваемого генератором случайных чисел. Время

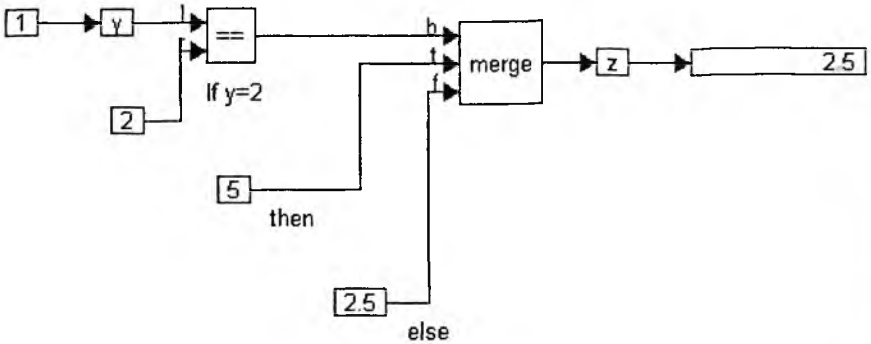


Рис. 3.50. Реализация конструкции if ... then ... else ... с помощью блока merge

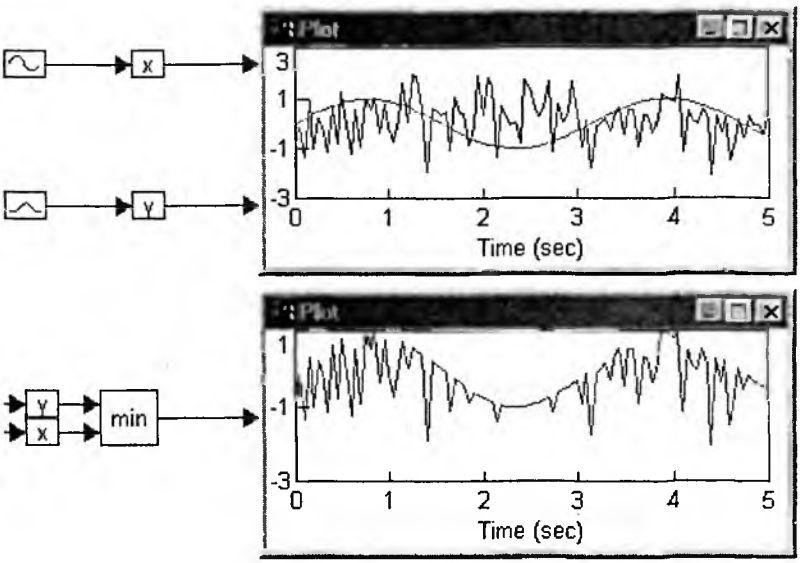


Рис. 3.51. Работа блока min

от времени тот или иной сигнал оказывается минимальным. Временные зави симости сигналов на входах блока min и на его выходе дают прекрасное пред ставление о характере работа блока.

Окно свойств блока min никаких настроек не содержит. Можно лишь за дать метку блока.

3.9.10. Блок квантования сигналов quantize

Блок квантования сигналов *квантует* сигнал по уровню с заданным ша гом (разрешением) по уровню Dy. В результате вместо непрерывного сигнал создается ступенчатый квантованный сигнал. Знак величины шага квантова ния смещает характеристику квантователя на величину шага, так что она рас

полагается сверху или снизу квантуемой зависимости. Работа блока иллюстрируется следующими соотношениями:

y = [целая_часть(x/Dy)] Dy — quantum,
quantum = Dy, если (sign(Dy) Чx < 0)
quantum = 0, иначе.
Здесь Dy — величина шага квантования.

Основное назначение блока — это моделирование аналогово-цифровых преобразователей. В окне свойств блока квантования можно задать величину шага квантования Dy (значение по умолчанию 0.05) и определить метку для блока. Пример применения блока quantize представлен на рис. 3.52.

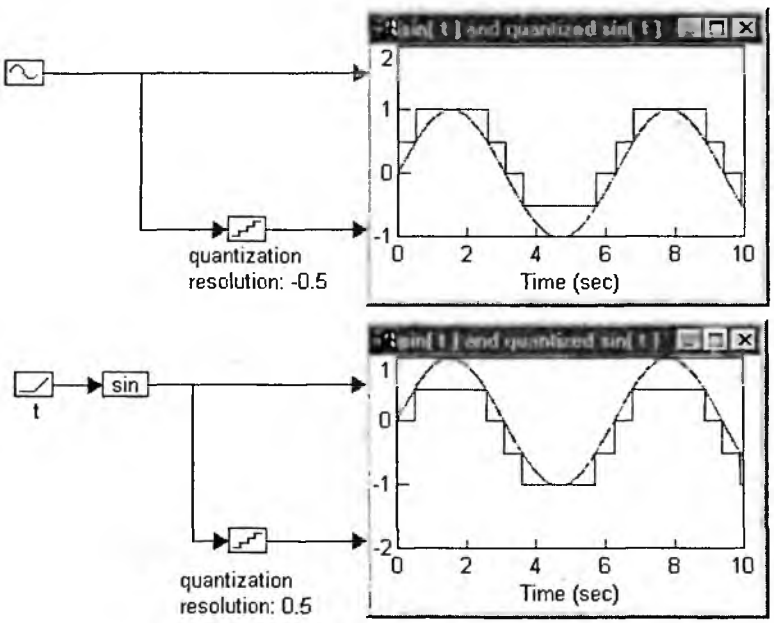


Рис. 3.52. Работа блока quantize

В цифровых системах квантование обычно предшествует применению аналого-цифровых преобразователей, преобразующих каждую ступеньку выходного сигнала квантователя в цифровой код.

3.9.11. Блок реле relay

Блок реле отслеживает величину входного сигнала на каждом шаге моделирования и в зависимости от заданной «зоны нечувствительности» устанавливает выход в одно из трех стабильных состояний:

y = –1, если x < – Dx/2
y = 1, если x > Dx/2
y = 0, иначе.

Здесь Dx — ширина зоны нечувствительности. Зона нечувствительности блока реле симметрична относительно нуля. В окне свойств блока можно задать ширину зоны нечувствительности Dx (значение по умолчанию 0) и определить метку для блока. Рисунок 3.53 иллюстрирует работу блока реле при подаче на его вход синусоидального сигнала.

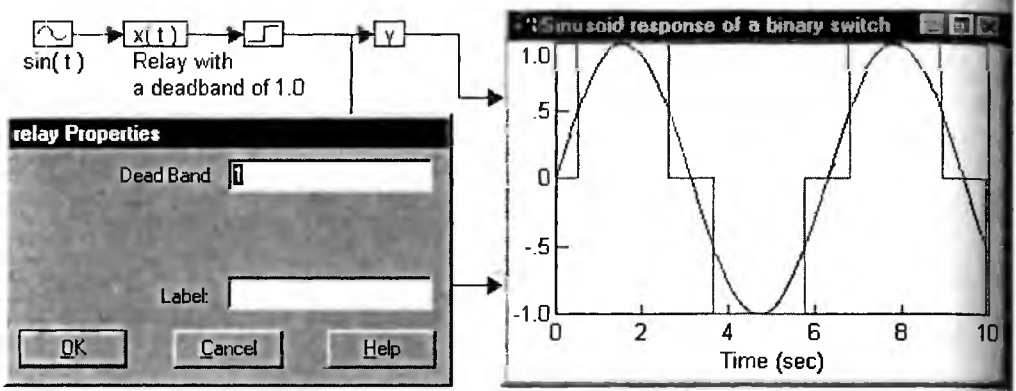


Рис. 3.53. Работа блока реле при синусоидальном входном сигнале

Если необходимо смоделировать компаратор, имеющий гистерезис, то можно охватить блок реле с нулевой шириной зоны нечувствительности положительной обратной связью. В канале обратной связи должен быть установлен любой малоинерционный элемент — например, блок регистра задержки или насыщаемый интегратор. Величина статического коэффициента передачи в канале обратной связи будет определять ширину петли гистерезиса.

3.9.12. Блок фиксатора sampleHold

Блок фиксатора sampleHold предназначен для запоминания мгновенного значения входного сигнала в момент поступления управляющего сигнала с сохранением значения выборки до следующего управляющего сигнала. Блок имеет логический пороговый управляющий вход, на который подается управляющий сигнал xb. Если модуль управляющего сигнала больше или равен 1, то блок работает в режиме выборки (повторяет входной сигнал). Если модуль управляющего сигнала меньше 1, то блок работает в режиме хранения. Таким образом, блок реализует работу в соответствии со следующими выражениями:

y [шаг] = y [шаг – 1], если | xb | < 1
y = x2, иначе.

Данная операция называется также *экстраполяцией нулевого порядка*. Она часто используется для запоминания мгновенного значения аналогового сигнала для последующей его оцифровки аналогово-цифровым преобразователем. Пример работы блока представлен на рис. 3.54.

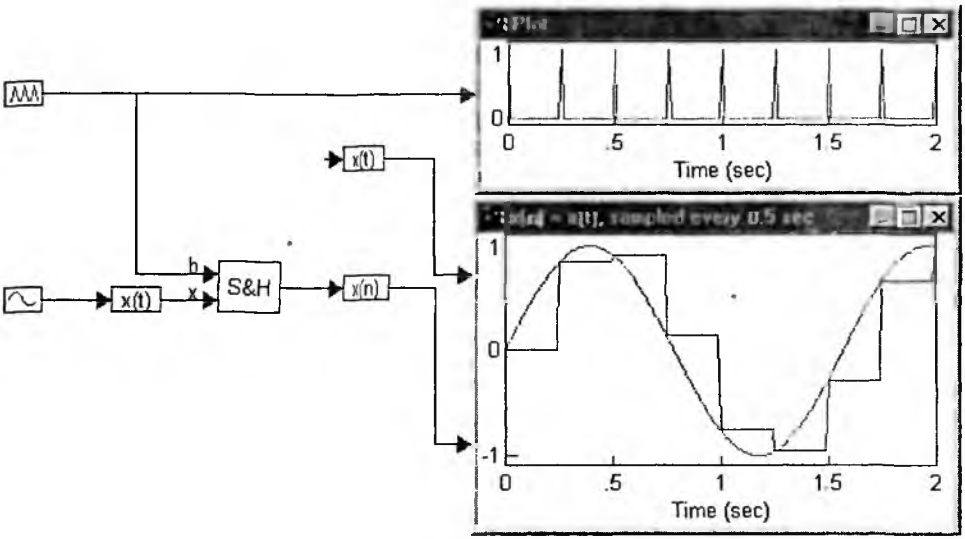


Рис. 3.54. Пример работы блока хранения выборки

Диалоговое окно свойств блока позволяет установить начальное хранимое значение (по умолчанию равно 0) и определить метку для блока.

3.10. Блоки оптимизации

3.10.1. Подготовка к оптимизации

Оптимизация модели заключается в таком подборе ее параметров, при котором обеспечивается минимум некоторой целевой функции, например определяющей значение каких-либо параметров. Оптимизация может использоваться для поиска корней нелинейных уравнений или оптимизации усилителя с коррекцией по минимуму времени установления сигнала на выходе.

Оптимизация завершенной модели является целью многих задач моделирования. В VisSim она реализуется особым построением модели, в которую включаются специальные блоки для многократного моделирования с изменяемыми параметрами. Изменения параметров производятся итерационно до получения минимума некоторой целевой функции системы. Задачей пользователя является подготовка модели таким образом, чтобы целевая функция обеспечивала достаточно хорошо выраженный минимум.

Нередко встроенным в VisSim оптимизаторам удается найти глобальный минимум целевой функции. Но, в общем случае многоэкстремальных целевых функций глобального минимума оптимизация найти не может. Решение может сойтись к локальному минимуму. Однако изменением параметров оптимизации можно найти имеющиеся минимумы (если их не слишком много) и затем определить, какой же из них является глобальным.

Установку общих параметров оптимизации можно выполнить из окна установок оптимизации, которое выводится командой Optimization Properti-

es... в позиции Simulation меню. В этом окне (ввиду очевидности оно не приводится) можно выбрать метод оптимизации:

Powel — *безградиентный метод Поувелла*, в котором вычисления производных выполняются по упрощенным разностным формулам, что обеспечивает повышенную скорость оптимизации;

Polak-Rabiere — *градиентный метод Полака—Райбера*, оптимизированный под поиск минимума функций, близких к квадратичным зависимостям в окрестности точки минимума;

Fletcher Reeves — *градиентный метод Флетчера—Ривеса*, имеющий лучшую сходимость, чем метод Полака—Райбера, но несколько меньшую скорость поиска;

User Method — оптимизация *методом пользователя* (если, конечно, пользователь в состоянии разработать свой метод оптимизации).

Для включения оптимизации надо задать опцию Perform Optimization (Начать оптимизацию). В окне установок параметров оптимизации можно также задать максимальное число итераций Max Iterations и допуск ошибки Error Tolerance (разность значений целевой функции на двух последних итерациях). По умолчанию эти параметры равны соответственно 50 и 1. Задание большего Error Tolerance (например, 5 или 10) позволяет уменьшить число итераций и полезно в случаях, когда оптимизаторы «зацикливаются». При этом находится грубое решение, которое стоит использовать для более точного поиска минимума.

3.10.2. Блок задаваемых ограничений constraint

Блок задаваемых ограничений constraint служит для задания при оптимизации необходимых ограничений. Он используется решателями, задающими решение итерационными методами уравнений без использования производных (неявных уравнений). Блок используется в паре с блоком unknown (см. ниже). Этот блок, как и другие блоки оптимизации, может использоваться для решения уравнений, например импликативных, и решения задач оптимизации.

3.10.3. Блок задания целевой функции cost

Блок cost служит для задания целевой (оптимизируемой) функции. Он используется оптимизаторами системы VisSim, обеспечивающими автоматический подбор параметров системы в итерационном процессе. Блок используется в паре с блоком parameterUnknown. Целевую функцию следует составлять в соответствии с характером решаемой задачи.

3.10.4. Блок глобальных ограничений globalConstraint

Блок globalConstraint предназначен для создания связи с собственным оптимизатором пользователя, написанным на языке высокого уровня Си++, Паскаль, Фортран и других, и откомпилированным в dll-библиотеку. Применительно к задачам данной книги он нигде не используется.

3.10.5. Блок задания неизвестных parameterUnknown

Блок parameterUnknown (неизвестный параметр) используется оптимизаторами для подстановки в блок-схему подбираемых ими в итерационном процессе повторном моделировании со значениями параметров системы, минимизирующими целевую функцию. Блок используется в паре с блоком-датчиком целевой функции.

3.10.6. Блок задания неизвестной unknown

Блок unknown (неизвестная) используется неявными решателями для подстановки в блок-схему начальных значений неизвестных величин. Блок используется в паре с блоком constraint. Цель применения этой пары — решение уравнений, не содержащих производных (неявных уравнений). Условием возможности получения численного решения неявного уравнения является прямая или косвенная зависимость значения сигнала на входе блока (или блоков) constraint от выходных значений всех введенных блоков типа unknown.

3.10.7. Пример решения нелинейного уравнения

Нелинейные уравнения могут решаться с применением описанных выше средств. Напомним, что решением нелинейного уравнения является нахождение его корней, т. е. значений независимой переменной, при котором уравнение обращается в нуль. Поскольку нуль — это наименьшее значение, то очевидна возможность нахождения решений нелинейных уравнений методом минимизации (оптимизации).

В качестве примера возьмем уравнение $y + 5 \cdot \cos(y) = 0$. Чтобы найти приближенно решения, построим график этой зависимости. Это сделано на рис. 3.55 сверху. Нетрудно заметить, что наше уравнение имеет три решения при значениях y близких к –1.3, 2 и 3.8. Под графиком рис. 3.55 представлена типичная модель для решения подобных уравнений. В ней слева задается начальное значение независимой переменной y с помощью блока unknown. За-

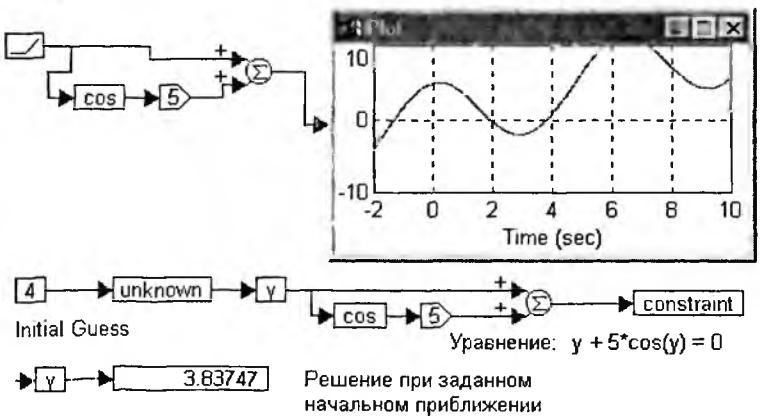


Рис. 3.55. Модель решения уравнения $y + 5 \cdot \cos(y) = 0$

тем формируется выражение, представляющее левую часть уравнения, и вы
ход выражения подключается к блоку решения constraint.

При пуске модели начинается итерационный процесс минимизации зн
чения $y + 5 \cdot \cos(y)$, и по его завершении выводится полученное значение пе
ременной y при $y + 5 \cdot \cos(y) = 0$.

Задав начальные значения равными 2 и 4, можно найти еще два решени
при $y = 1.97738$ и $y = 3.83747$. Естественно, что после каждой смены начально
го значения переменной y надо снова запускать модель.

Здесь уместно отметить, что в системе Mathcad решение подобного ура
нения выполняется еще проще, например с применением функции root:

$y := -2 \quad root(y + 5\cos(y), y) = -1.30644$

$y := 2 \quad root(y + 5\cos(y), y) = 1.97722$

$y := 4 \quad root(y + 5\cos(y), y) = 3.83747$

Поэтому решение подобных уравнений в среде VisSim рационально тол
ко в случае отсутствия подходящей системы компьютерной математики и ко
да необходимо унифицировать средства решения, например как в приведен
ном выше примере.

3.11. Блоки генераторов шума

3.11.1. Блок задания Гауссова шума gaussian

VisSim позволяет легко задавать детерминированные сигналы, наприме
синусоидальные или прямоугольные импульсы. Однако для полноценного
моделирования электро- и радиотехнических устройств необходимо учитыва
шумы, которые возникают в электронных компонентах или присутствуют
каналах связи (см. главу 1). Для этого VisSim имеет три блока генератор
шума с различными законами распределения случайных чисел, которыми эт
шумы создаются.

Блок задания Гауссова шума gaussian генерирует псевдослучайный шум
вой сигнал с нормальным распределением (см. главу 1) и задаваемыми пар
метрами: средним значением Mean и стандартным отклонением Standard Dev
vation (корень квадратный из дисперсии). По умолчанию они равны 0 и 1 со
ответственно. Эти параметры устанавливаются в окне свойств данного блока.

При повторении симуляции вся псевдослучайная последовательность по
вторяется, т. е. значения ее выборок для n-го шага симуляции неизменны
Для смены последовательности на вкладке Preferences окна свойств моделиро
вания Simulation Properties надо задать новое число Random Seed, инициир
ющее последовательность случайных чисел.

Для просмотра шума удобно использовать блок графопостроителя Plo
(рис. 3.56). Нетрудно заметить, что Гауссова шум представляет собой случай
ную зависимость при параметре Mean = 0, колеблющуюся случайным образо
вокруг нулевого значения. Вероятность больших значений такого шума быст
ро падает, и значения шума выше трех по модулю мало вероятны.

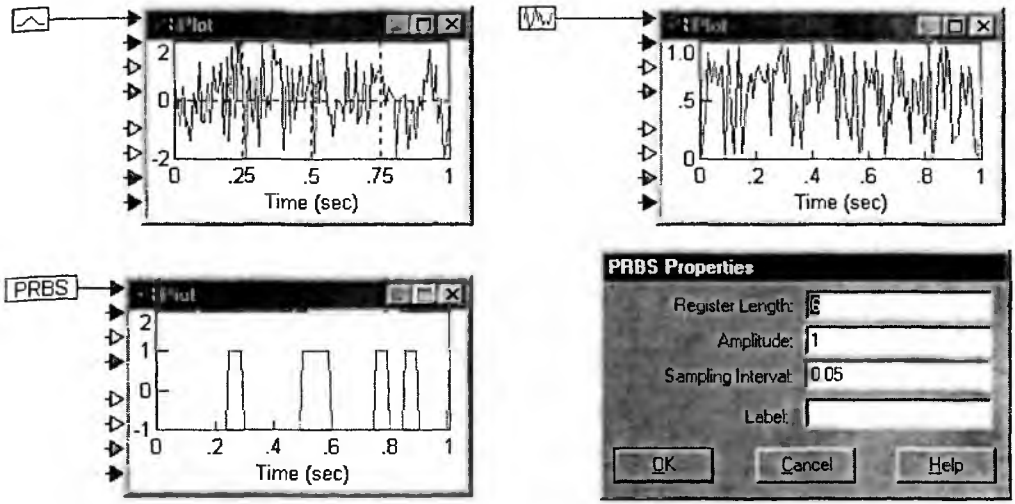


Рис. 3.56. Примеры просмотра шума, создаваемого блоками генераторов шума

3.11.2. Блок генератора шума с равномерным распределением uniform

Блок uniform генерирует псевдослучайный шумовой сигнал с равномерным распределением выборок на интервале от 0 до 1. Среднее значение такого шума равно 0.5, так что сделать его нулевым можно, добавив к шуму константу –0.5. Все сказанное о псевдослучайности Гауссового шума относится и к данному виду шума. Пример просмотра этого вида шума дан на рис. 3.56. В окне свойств помимо установки метки блока есть возможность задать временную задержку Time Delay в секундах (по умолчанию она задана нулевой).

3.11.3. Блок задания псевдослучайного бинарного сигнала PRBS

Блок задания псевдослучайного бинарного сигнала PRBS генерирует импульсный сигнал постоянной величины, знак которого меняется псевдослучайно и с равной вероятностью. Генератор построен на основе регистра сдвига. В окне свойств этого блока (см. рис. 3.56) можно задать длину регистра генератора (период повторения последовательности) Register Length, амплитуду Amplitude и период очередной выборки Sampling Interval. При повторении симуляции регистр сдвига генератора обнуляется. Если количество разрядов регистра задано равным N, то псевдослучайная последовательность повторится через 2N – 1 выборку. Значение по умолчанию — 6 (повтор через 63 выборки). По умолчанию заданы параметры Amplitude = 1 и Sampling Interval = 0.05.

Блок PRBS часто используется для изучения влияния случайных воздействий на дискретную систему. Он также используется при решении задач идентификации систем, т. е. при создании их математических моделей на основе измерительной информации.

3.12. Раздел виртуальных приборов и датчиков

3.12.1. Блок цифрового индикатора display

Блок display отображает текущие значения входного сигнала (рис. 3.57). В окне свойств дисплея можно изменить показание Value, задать нужное число знакомест (от 6 до 15), изменить цвет цифр и фона, задать экспоненциальное представление корней.

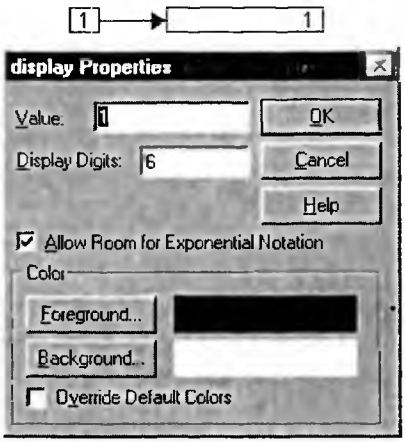


Рис. 3.57. Блок дисплея и окно его свойств

При подключении блока к шинному проводнику дисплейная строка размножается автоматически согласно конфигурации шины. Данные отображаются в асинхронном режиме, поэтому процесс симуляции не затормаживается.

3.12.2. Блок ошибки error

Блок ошибка осуществляет проверку входного сигнала и в случае, если он не равен нулю, останавливает процесс моделирования на заданном шаге, устанавливает флаг ошибки и раскрашивает блок, а также содержащие его составные блоки красным цветом. Сбросить флаг ошибки можно щелчком другой кнопки мыши по блоку или командой Clear Errors в позиции Edit меню (последняя сбрасывает флаги и красную окраску при ошибке и для других блоков). Примеры применения блока error представлены на рис. 3.58.

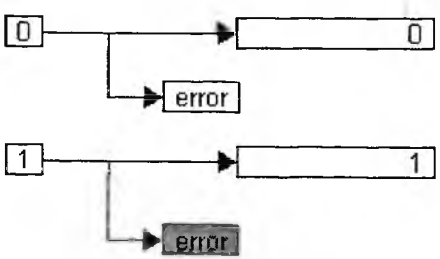


Рис. 3.58. Примеры применения блока ошибки

Разумеется, этот блок вовсе не предназначен просто для фиксации отклонения сигнала от нуля. Реальное его применение предполагает наличие некоторой логической (анализирующей) части модели, сигналы которой и являются входными для блоков ошибки.

3.12.3. Блок экспорта сигналов в файл export

Блок export осуществляет запись сигналов, поступающих на его входы в файл. Число входов у блока изначально равно трем, но его можно уменьшить или увеличить. Это число задает количество каналов записи. Пример применения блока для записи вектора в файл данных demo с расширением .bat дан на рис. 3.59. После записи имя блока export заменяется названием файла.

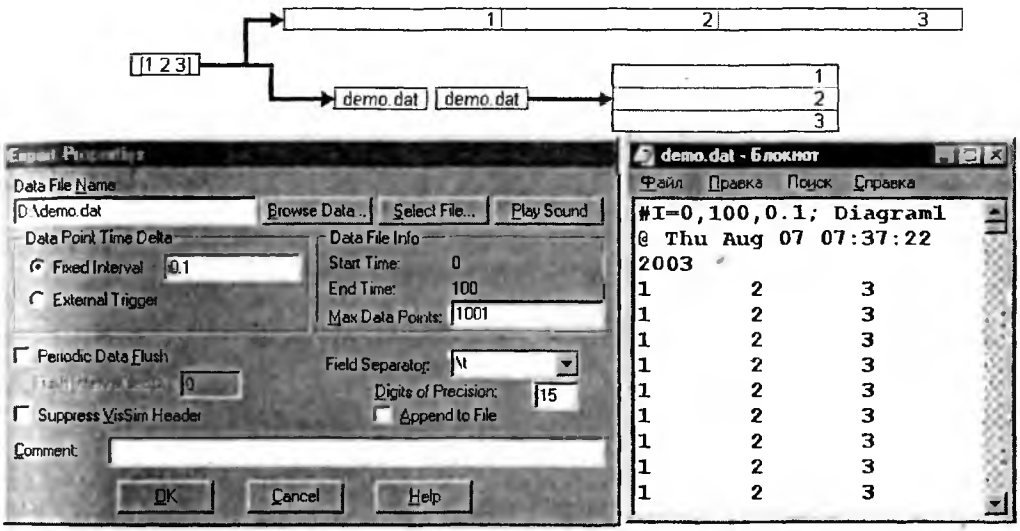


Рис. 3.59. Применение блока export и окно его свойств

В соответствии с типом данных можно выбрать одно из стандартных расширений файлов: .dat, .m, .mat или .wav. Впоследствии данные могут быть импортированы для обработки либо в VisSim, либо в другие программы — MATLAB, Microsoft Excel и др. Кнопка Browse Data... открывает окно редактора (обычно это Блокнот), в котором можно просмотреть файл и уточнить его формат. Для звуковых файлов их можно прослушать с помощью кнопки Play Sound. Данные могут поступать с фиксированным интервалом времени или в соответствии с работой внешнего триггера. С другими деталями окна свойств можно ознакомиться выведя его, что и показано на рис. 3.59. Этот блок может использоваться совместно с блоком импорта данных import.

3.12.4. Блок построения гистограмм histogram

Блок «гистограмма» визуализирует вероятностное распределение потока данных, поступающих на вход блока во время моделирования. Это позволяет судить, например, о характере и типе распределения данных. На рис. 3.60 представлен пример построения гистограммы для данных, полученных от

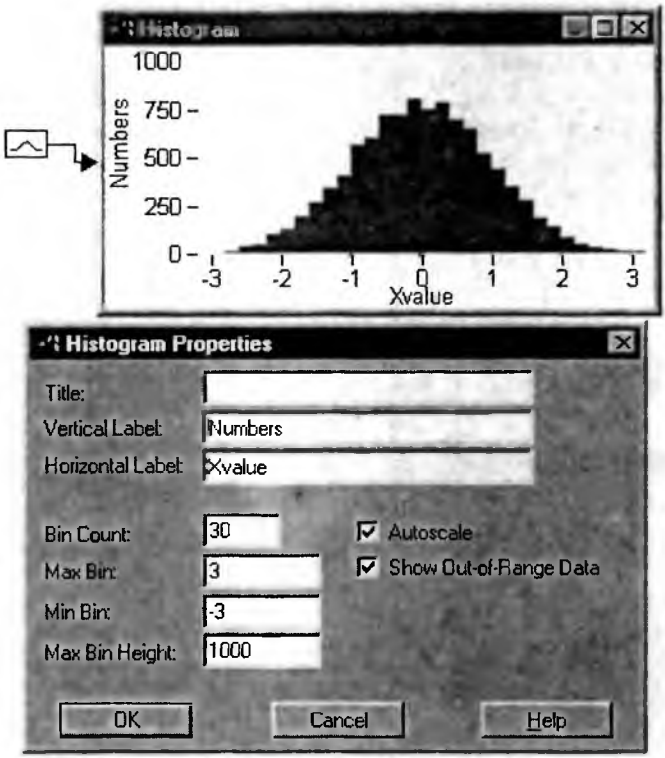


Рис. 3.60. Гистограмма набора из 10 000 случайных чисел с законом распределения Гаусса

блока генератора случайных чисел с Гауссовы распределением. Нетрудно з метить, что пик гистограммы наблюдается в районе X = 0, а затем частоты п падания данных уменьшаются в обе стороны от X = 0, падая почти до 0 п X = –3 и X = 3. Это хорошо соответствует представлениям о распределен случайных чисел по закону Гаусса.

Чтобы гистограмма была правильной, нужна настройка блока с помощ окна его свойств, представленного на рис. 3.60. В окне можно задать офор ление титульной надписи (в использованной версии UisSim это оказалось возможным) и надписей по вертикальной и горизонтальной оси. Далее мож выбрать число столбцов гистограммы Bin Count, масштабы по горизонт (Min Bin и Max Bin) и вертикали (Max Bin Height). Для первоначального п строения гистограммы удобно использовать автомасштабирование — опц Autoscale.

Рисунок 3.61 показывает построение гистограммы распределения случ ных чисел с равномерным распределением, создаваемых блоком uniform. же показано окно свойств с настройкой под построение представленной г тограммы.

Сравнение гистограмм рис. 3.60 и 3.61 дает наглядное представление различии распределения случайных чисел, создаваемых блоками gaussian uniform.

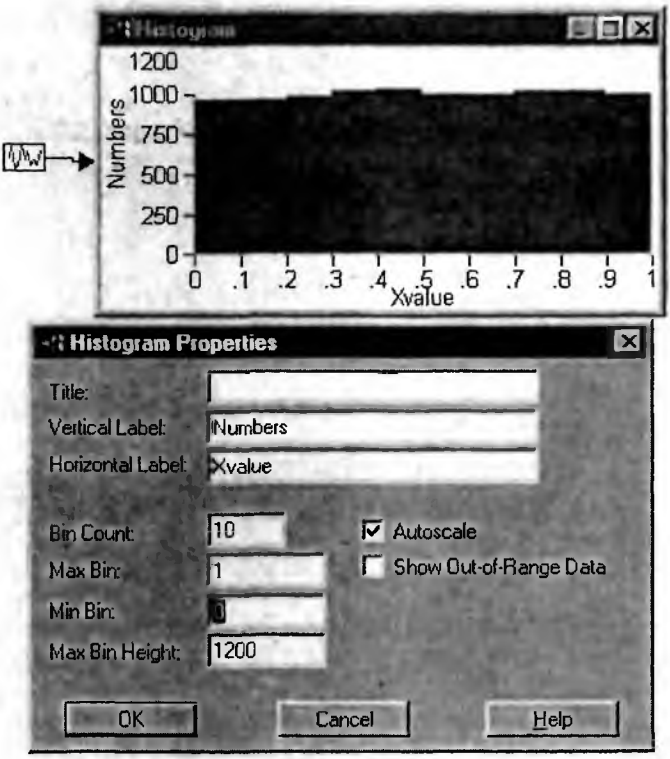


Рис. 3.61. Гистограмма набора из 10 000 случайных чисел с равномерным законом распределения

3.12.5. Блок световой индикации light

Блок световой индикации имитирует работу светодиодного индикатора или пробника. Он отслеживает нахождение входного сигнала в трех диапазонах, которые задаются двумя предельными значениями (уровнями) — верхним ub и нижним lb. Текущее значение входного сигнала заставляет блок светиться синим, зеленым или красным цветом в соответствии с выражениями его работы:

y = • (красный), если x > ub
y = • (зеленый), если lb ≤ x ≤ ub
y = • (синий), если x < lb

Цветовая сигнализация может быть заменена тремя графическими изображениями с соответствующим содержанием. Каждое состояние блока может сопровождаться звуковым сигналом или сообщением. Верхний ub и нижний lb предельные уровни задаются в диалоговом окне свойств блока, и нет возможности динамически менять их значения. Блок обрабатывает только отдельные сигналы (нельзя подключить шинный проводник к входу). Пример применения блока light представлен на рис. 3.62. Там же показано окно свойств блока.

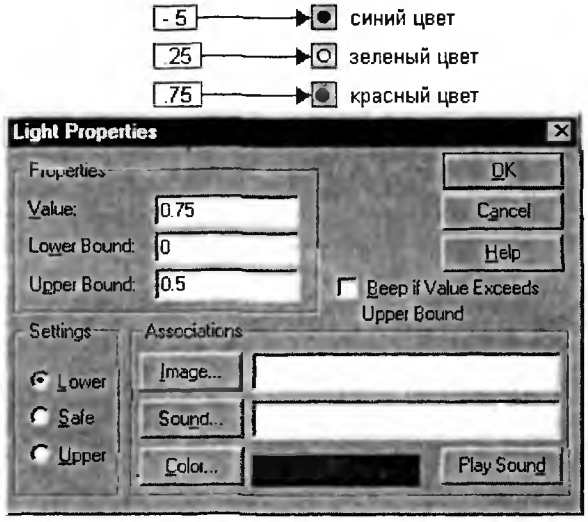


Рис. 3.62. Пример применения блока light и окно его свойств

Нажатие на кнопку Image окна свойств данного блока вызывает открыт системного диалогового окна выбора графического файла формата .bmp, зад ющего рисунок, который соответствует данному диапазону входного сигнал Ручной ввод пути к файлу и его имени в располагающуюся рядом строку я ляется альтернативным способом подключения изображения. Обычно нуж повторить операцию трижды, для каждого диапазона входного сигнала.

Кнопка Sound... позволяет загрузить звуковой файл, который можно ис пользовать для сигнализации (если, конечно, компьютер оснащен звуковым средствами). Кнопка Color... вызывает цветовую палитру, позволяющую уст новить и другие цвета сигнализации помимо отмеченных выше. Наконе кнопка Play Sound позволяет воспроизвести выбранный звуковой файл. В эти настройки могут осуществляться для каждого из трех отмеченных вы диапазонов.

3.12.6. Блок стрелочного измерителя meter

Блок meter представляет величину входного сигнала, отображая ее с п мощью стрелочного или шкального указателя (рис. 3.63).

Этот блок прибор только визуализирует сигнал и не является модель электромеханических преобразователей, к коим относятся широко распр страненные измерительные приборы, такие как магнитоэлектрические, элек ромагнитные или электродинамические. Другими словами, как и другие и дикаторные и измерительные блоки, данный прибор является виртуальным.

При добавлении входов блок отображается как приборная панел (рис. 3.64).

Блок meter может отображать меняющийся сигнал, например от источн ка синусоидального сигнала. Однако при быстром изменении сигнала движ ния стрелки могут оказаться незаметными или плохо заметными.

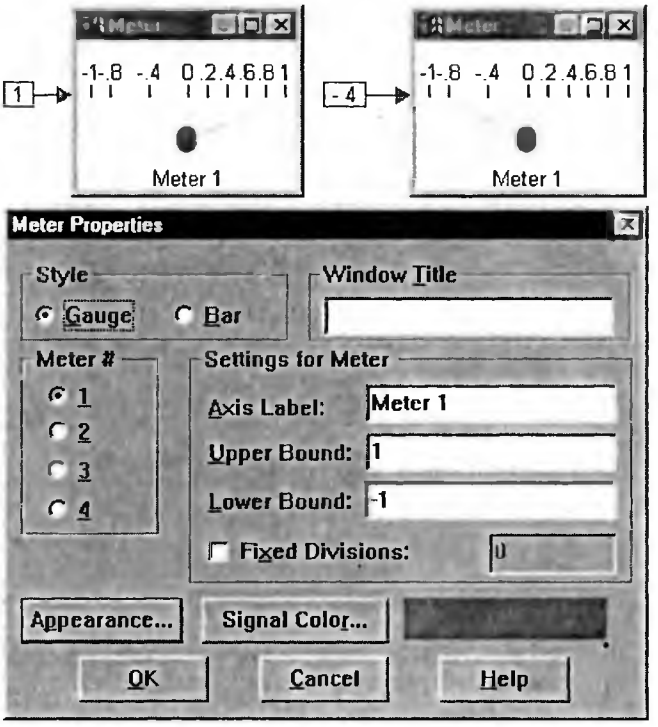


Рис. 3.63. Примеры применения блока meter для измерения сигналов-констант

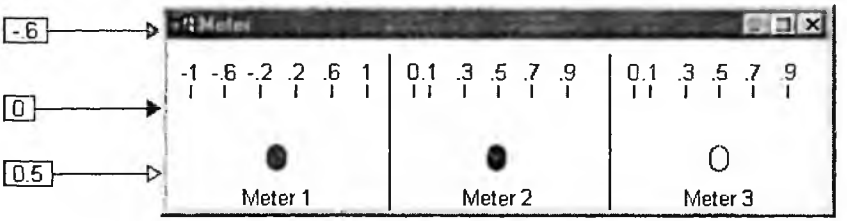


Рис. 3.64. Создание из блока meter приборной панели

3.12.7. Блок графопостроителя (осциллографа) plot

Блок виртуального графопостроителя (осциллографа) является наиболее удобным виртуальным прибором для визуального отображения сигналов и результатов моделирования. Мы уже неоднократно использовали этот блок с этой целью, так что рассмотрим его уточненные настройки, которые доступны из окна свойств (рис. 3.65). Это окно имеет 5 вкладок, и на рис. 3.65 открыта первая из них — Options (Опции).

Вкладка Options является основной и задает множество возможностей и параметров графопостроителя.

Опция Fixed bound позволяет зафиксировать масштабы по осям X и Y, установленные на вкладке Axis. Это часто полезно, поскольку по умолчанию графопостроитель устанавливает масштабы автоматически, и они не всегда

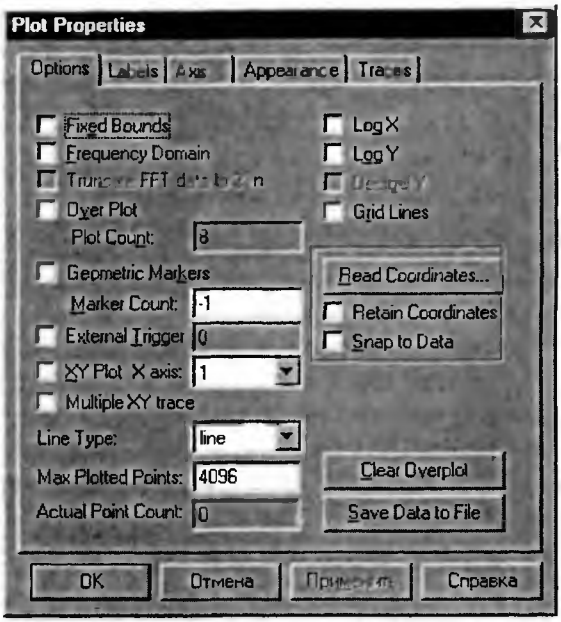


Рис. 3.65. Окно свойств блока plot с открытой вкладкой Options

удобны. Например, при отображении синусоиды с единичной амплитудо масштаб по вертикали устанавливается равным точно –1 и +1. Это не всег приемлемо, поскольку верхушки синусоиды будут касаться верхней и нижне границ рабочей части окна графопостроителя, что создает эффект их подреза ния. Более подходящими будут значения –1.1 и 1.1. При этом верхушки сину соиды отображаются «во всей своей красе».

Опции Log X и Log Y предназначены для переключения между линейны и логарифмическим масштабами представления данных по горизонтали вертикали. Активизация опции соответствует включению логарифмическог масштаба по соответствующей оси. Однако в этом случае невозможно отобр зить данные с отрицательными значениями. При их появлении в исходно совокупности выборок произойдет отсечение графика в декаде, в которую по падет наименьшее положительное значение сигнала.

Опция Y в [дБ] в случае логарифмического масштаба задает построени меток по оси Y (20*log(Y)). Если значение в логарифмическом масштабе из менилось на 20 дБ, то в линейном масштабе это соответствует изменению 10 раз. Опция оказывает влияние на вид масштабной сетки при ее включени В случае логарифмического масштаба она становится неравномерной.

Опция Geometric Markers строит кривые вместе с маркерами в форме раз личных геометрических фигур, которые устанавливаются на вкладке Trace Вы можете указать число маркеров, причем 0 означает вывод всех маркеро Подобное оформление полезно для распечаток на черно-белом принтере при визуализации на монохромном дисплее, когда по цвету кривые неразл чимы.

Опция Over plot приводит к сохранению указанного количества после них следов Plot points осциллограмм при повторных пусках моделировани

Это дает возможность отследить действие вариации параметров на модель. Кнопка Clear Overplot приводит к очистке всех следов осциллограмм.

Активизация опции External Trigger приводит к появлению у осциллографа дополнительного входа круглой формы, с помощью которого может быть организована внешняя синхронизация логическим сигналом — 0 или 1. Стробирование данных (регистрация и вывод на дисплей) осуществляется, если сигнал синхронизации равен единице (1).

Опция Line type (Тип линии) используется для установки из списка типа линий. Можно задать построение графиков различными типами линий — непрерывными, точечными и ступенчатыми.

Опция Max Plotted Points задает число точек (выборок) каждой кривой. Максимальное число регистрируемых и отображаемых на экране выборок может составлять 250 миллионов. Выборки могут регистрироваться столь плотно, что соседние при визуализации будут попадать в один пиксель экрана. В подобных случаях вы можете вести регистрацию с прореживанием. Тогда все операции перерисовки и печати осциллограммы будут существенно ускорены. В большинстве случаев достаточно около 100 точек, если только не требуется последующее изучение данных с увеличением масштаба. Если в строке ввода ввести цифру ноль (0), то режим прореживания будет отключен. Затемненное поле Actual Point Count отображает количество выборок, которые составили осциллограмму.

Опция Grid Line включает построение координатной сетки. Интервал между линиями сетки устанавливается автоматически в зависимости от диапазонов визуализации и размера графика. Но возможно определить сетку принудительно, на вкладке Axis, описанной ниже.

Кнопка Save Data to File вызывает открытие файлового диалогового окна с возможностью уточнения имени и типа файла для записи зарегистрированных осциллографом данных. Раскройте в диалоговом окне выпадающий список для выбора требуемого формата сохранения

Вкладка Labels (Метки) служит для установки различных надписей — меток. Она представлена на рис. 3.66. По умолчанию задана лишь одна из надписей по оси X. Можно также задать надпись по оси Y, титульную надпись, подтитульную надпись и надписи для обозначения кривых.

Вкладка Axis (Оси) служит для форматирования осей (рис. 3.67). Здесь можно установить масштабы по осям X и Y и время Time Scaling. В области Axis Divisions можно задать деление осей и вывод реперных меток по осям. Обычно временной масштаб блока задается автоматически, но, задав опцию Retrace Enable, можно установить вручную начальное время Start Time, конечное время Ens Time и временной интервал Interval.

Вкладка Appearance (Внешний вид), представленная на рис. 3.68, позволяет изменять цветовое оформление окна блока Plot. Кнопки Foreground... и Background... выводят палитры цветов для установки цветов переднего и заднего фронтов. Поле Bitmap позволяет вместо цвета задать изображение из графического файла. Выбор файла производится из окна, вызываемого кнопкой Image... Опция Override default colors задает возврат к исходным цветам, принятым по умолчанию.

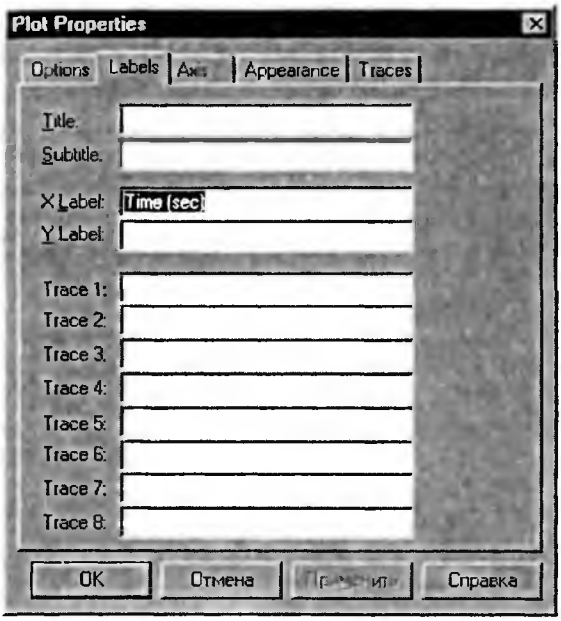


Рис. 3.66. Окно свойств блока plot с открытой вкладкой Labels

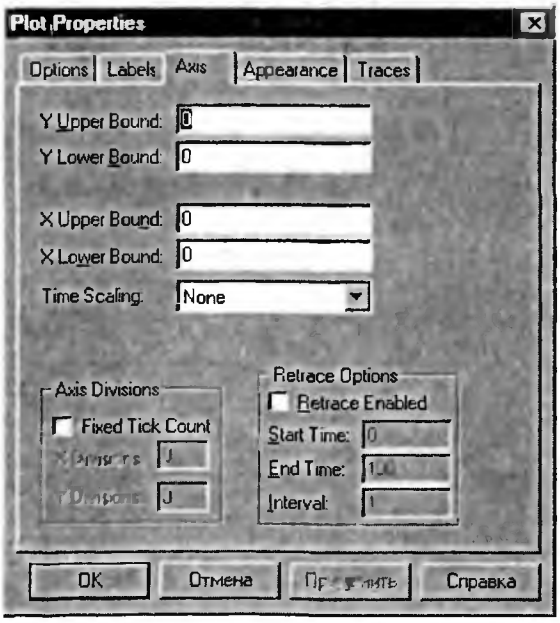


Рис. 3.67. Окно свойств блока plot с открытой вкладкой Axis

Наконец вкладка Traces (Кривые), представленная на рис. 3.69, позволя изменить цвет каждой линии графика и задать вид меток на этих лини (если предусмотрен их вывод). Для каждой из обозначенных номерами кр

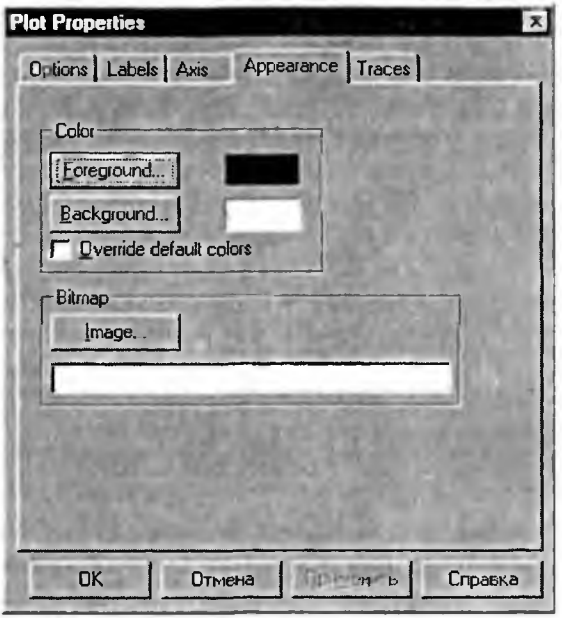


Рис. 3.68. Окно свойств блока plot с открытой вкладкой Appearance

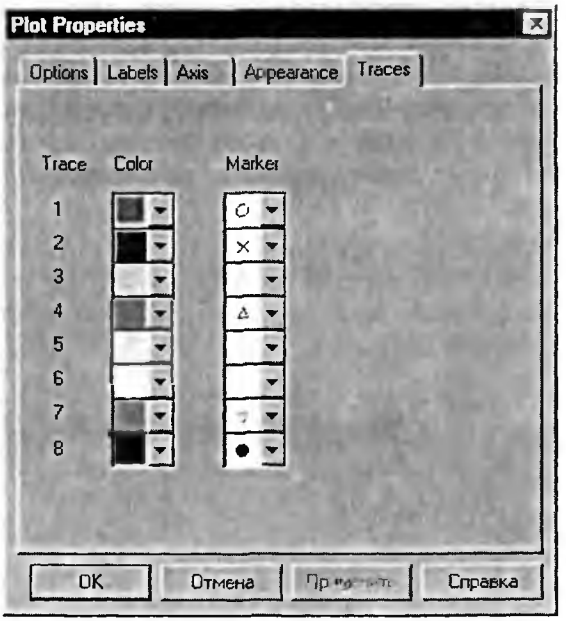


Рис. 3.69. Окно свойств блока plot с открытой вкладкой Traces

вых выводится свой список цветов и список меток. Из них и осуществляется выбор нужных цветов и нужных меток.

3.12.8. Детальный просмотр и печать осциллограмм

Блок виртуального осциллографа отображается окном, которое имеет стандартные средства управления размерами. Используя кнопки в конце его титульной строки, можно увеличить размер окна осциллографа до максимального, свернуть окно или закрыть и удалить его. Если требуется изменить окно плавно, то следует подвести указатель мыши к границе окна (или к его углам) и после смены вида курсора мыши на перемещающие стрелки выполнить коррекцию размера.

По умолчанию блок выводится с 8 входами, обозначенными треугольниками разного цвета. Но число входов можно уменьшить командой Remove Connector... или, напротив, увеличить командой Add Connector... . Эти команды находятся в позиции File меню. Подключение входов к нужным точкам или соединениям уже многократно описывалось. Остановимся на специальных применениям виртуального осциллографа, заметно расширяющего его возможности.

Начнем с реализации так называемой «лупы времени». Для детального ознакомления с фрагментом осциллограммы имеется возможность увеличить любую требуемую область дисплея. Регистрируемые виртуальным осциллографом данные имеют большую точность (тип double), и их число нередко составляет многие тысячи. Поэтому в отличие от реальных приборов сильное увеличение может иметь смысл.

Для применения «лупы времени» выполните следующие действия.

- Нажмите и удерживайте клавишу Ctrl.
- С помощью указателя мыши выделите требуемую область осциллограммы (нажав кнопку мыши и переместив указатель по диагонали). При этом в процессе перемещения указателя его текущие координаты в осциллограмме будут подсвечиваться цветом морской волны в левой нижней части окна осциллографа (рис. 3.70).
- Отпустите кнопку мыши — будет выполнен вывод в увеличенном масштабе выделенной части графика (рис. 3.71).
- Если требуется дополнительное увеличение, то повторите шаги 2 и 3.
- Отпустите клавишу Ctrl.

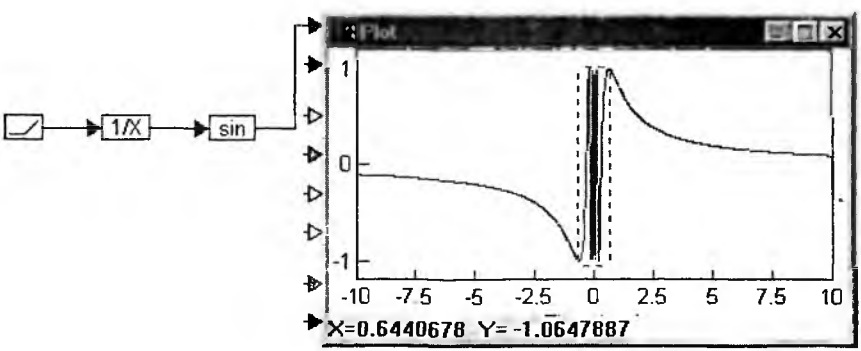


Рис. 3.70. Выделение фрагмента графика

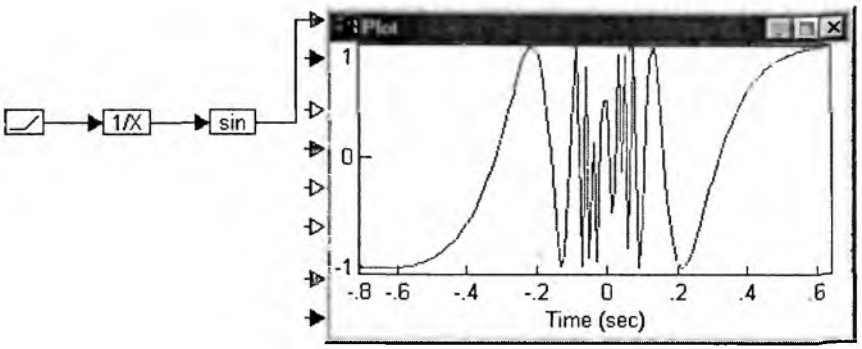


Рис. 3.71. Просмотр части графика в увеличенном масштабе

Для сброса осциллограммы к исходному масштабу выполните следующие действия.

- Нажмите и удерживайте клавишу Ctrl.
- Расположите указатель мыши в окне графопостроителя.
- Щелкните правой клавишей мыши, и масштаб будет сброшен к исходному.
- Отпустите клавишу Ctrl.

Для быстрого определения координат любой точки окна графопостроителя на вкладке Option (рис. 3.65) надо активизировать кнопку Read Coordinates... . На экране появится перемещаемое мышью зеленое перекрестие и поля для отображения координат X и Y центра перекрестия (рис. 3.72). Опция Snap to Data привязывает центр перекрестия к кривой, так что последний будет двигаться только по отображаемой кривой. Это позволяет существенно повысить точность анализа кривой, например определения координат ее особых точек. Для фиксации перекрестия используется клавиша Enter.

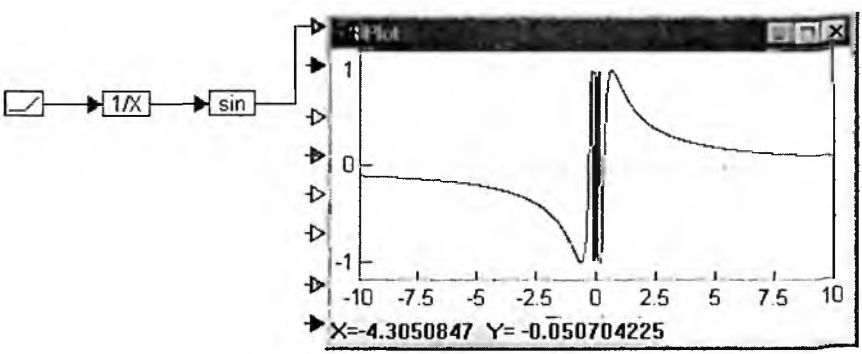


Рис. 3.72. Определение координаты точки окна графопостроителя

Стандартные команды печати блок-схемы модели имеются в позиции File меню. Но нередко требуется распечатка только окна осциллографа. Для распечатки осциллограммы выберите команду Print в системном меню окна осциллографа (верхний левый угол окна) (рис. 3.73).

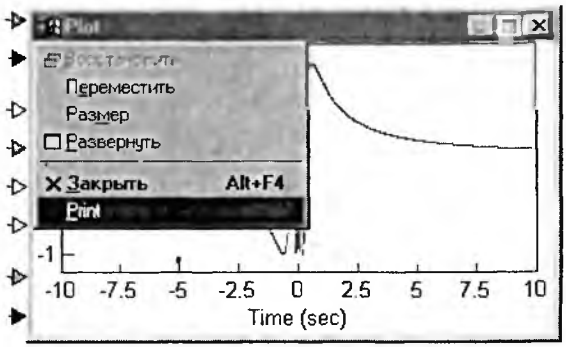


Рис. 3.73. Меню окна осциллографа

Исполнив эту команду, можно открыть окно печати осциллограммы. Оно показано на рис. 3.74 слева. Кнопка Setup в данном окне открывает стандартное окно установки параметров принтера — в нашем случае струйного принтера Epson Stylus Color 600. Заметим, что именно цветные струйные принтеры лучше всего подходят для печати цветных осциллограмм. Достаточно дешевые лазерные принтеры печатью в цвете не обладают.

Более детальное описание процесса печати не имеет смысла, поскольку оно ныне знакомо любому пользователю компьютером.

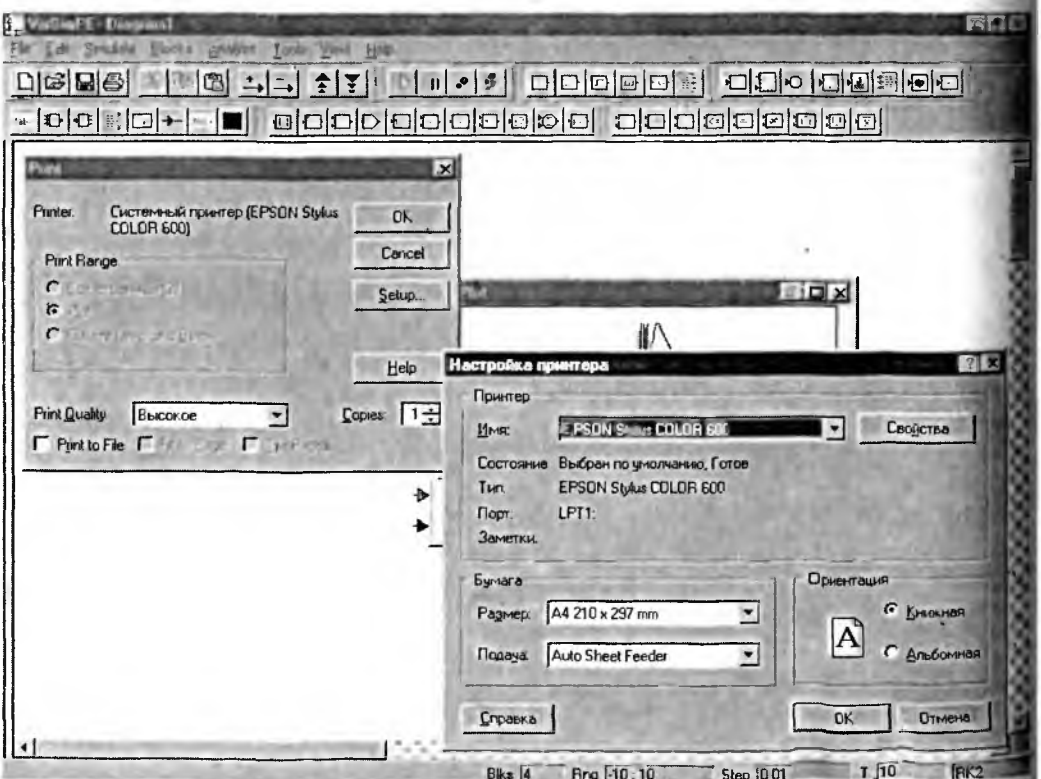


Рис. 3.74. Окна управления печатью графиков осциллографа

3.12.9. Построение спектра для графика данных в окне осциллографа

Опция Frequence Domain активизирует процесс выполнения операции быстрого преобразования Фурье для тех данных временного домена (для осциллограммы сигнала), которые остаются на дисплее после завершения процесса симуляции модели. Результат возвращается в виде спектра плотности мощности сигнала. Точность спектрального анализа связана с количеством зарегистрированных точек виртуальным осциллографом. Их количество по возможности должно быть кратно степеням двойки (8, 16, 32, 64, 128, ...), иначе спектр будет зашумлен.

Имеется несколько влияющих на работу осциллографа в качестве спектроанализатора факторов:

- Опция Truncate FFT data to 2^n будет обрезать последовательность выборок до соответствующего ближайшего количества. Если опция не активна, то последовательность данных дополняется нулями до количества, кратного ближайшей степени двойки. Эта опция может быть активирована, только когда активна опция Frequence Domain.
- Установка времени моделирования и шага для накопления требуемого количества отсчетов желательна исходя из условий выполнения спектрального анализа.
- Можно использовать линейный или логарифмический масштабы представления спектра.
- Опция Max Plotted Points может ограничить точность результата, но, как и внешняя синхронизация, обычно ускоряет вычисления.
- Если вы остановите процесс симуляции преждевременно, то точность выполнения операции спектрального анализа уменьшится.

Рисунок 3.75 показывает создание прямоугольного импульса с длительностью 0.1 с и его отображение в окне осциллографа при времени моделирования 1 с.

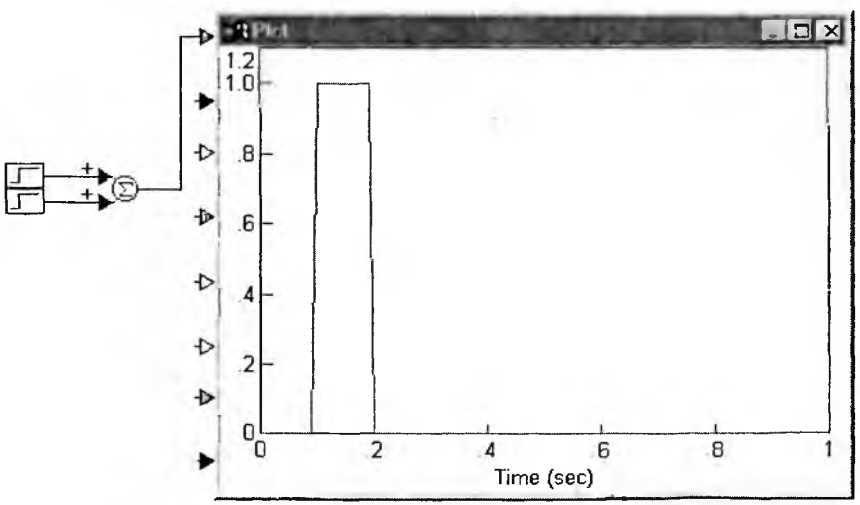


Рис. 3.75 Создание и отображение короткого прямоугольного импульса

Теперь для получения спектра гармоник этого импульса на вкладке Options зададим опцию Frequence Domain и при линейном масштабе по оси Y получим спектрограмму в виде, показанном на рис. 3.76.

Если установить масштаб логарифмическим, то спектр будет иметь вид, представленный на рис. 3.77. Здесь более ясно видны ничтожно малые амплитуды гармоник, попадающих на значения частот $n \cdot t_u$, где t_u — длительность импульса. Обычно логарифмический масштаб представления амплитуд гармоник позволяет лучше обнаруживать в спектре частотные составляющие даже с малой амплитудой.

Чудодейственное и практически мгновенное превращение виртуального осциллографа в виртуальный анализатор спектра по достоинству оценят все

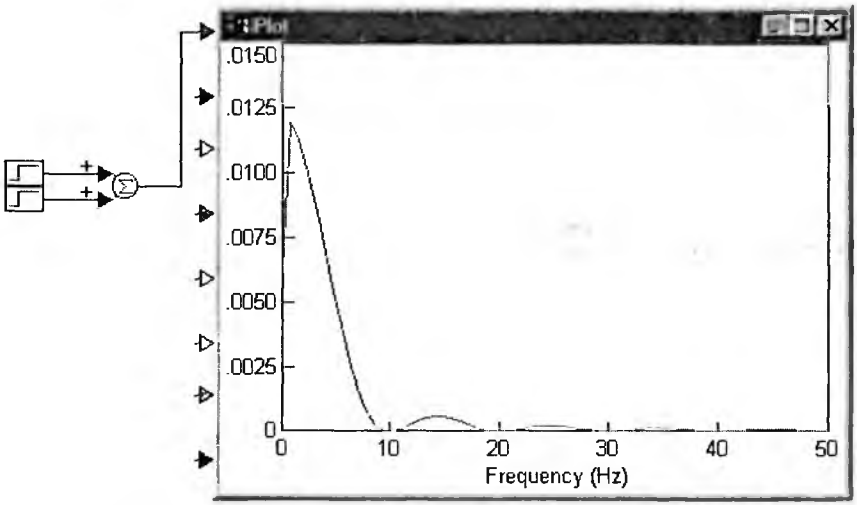


Рис. 3.76. Спектрограмма прямоугольного импульса при линейном масштабе отображения амплитуды гармоник

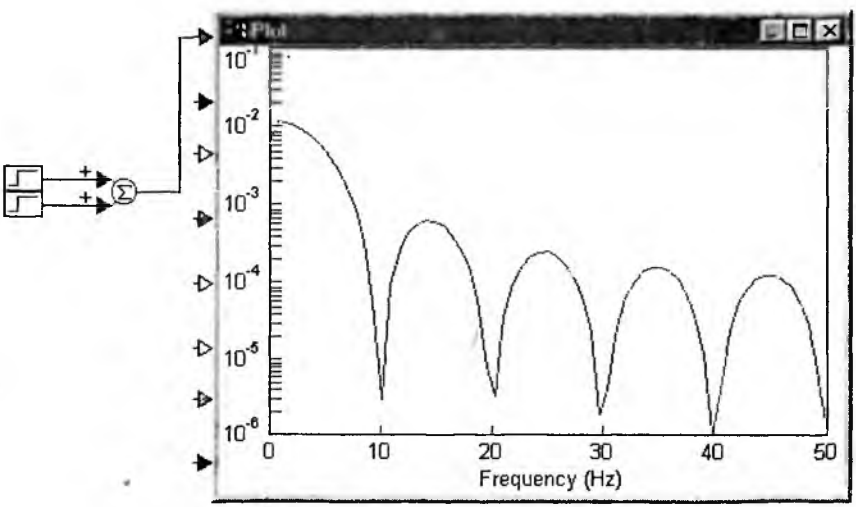


Рис. 3.77. Спектрограмма прямоугольного импульса при логарифмическом масштабе отображения амплитуды гармоник

пользователи, работающие с моделированием сигналов и устройств по обработке сигналов.

3.12.10. Представление фигур Лиссажу и фазовых портретов

Особое значение имеет также и опция XY Plot вкладки Options окна свойств виртуального осциллографа. Она позволяет подавать дополнительный сигнал на ось X, что нужно для построения фигур Лиссажу и фазовых портретов. Пример построения фигуры Лиссажу представлен на рис. 3.78 и 3.79.

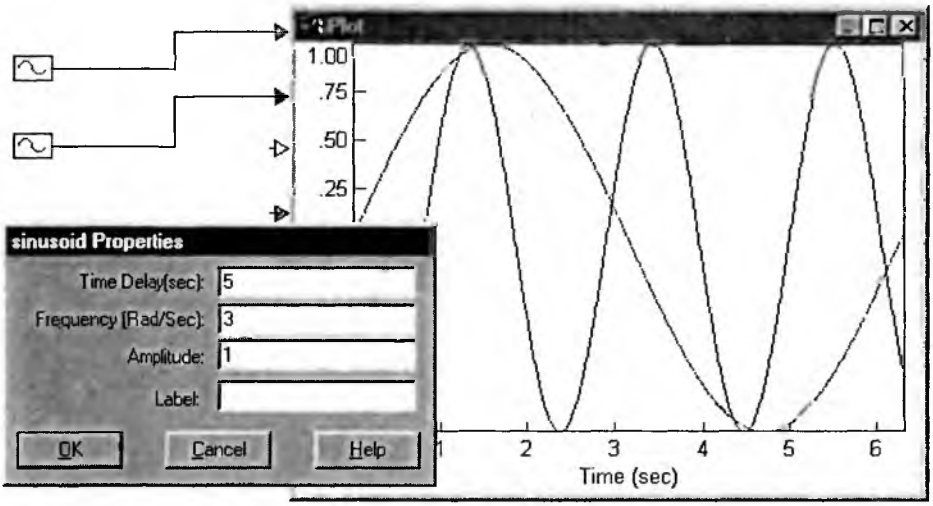


Рис. 3.78. Подготовка к построению фигуры Лиссажу — подключение к входам осциллографа двух генераторов синусоидальных колебаний — частота второго генератора втрое выше частоты первого и введена временная задержка

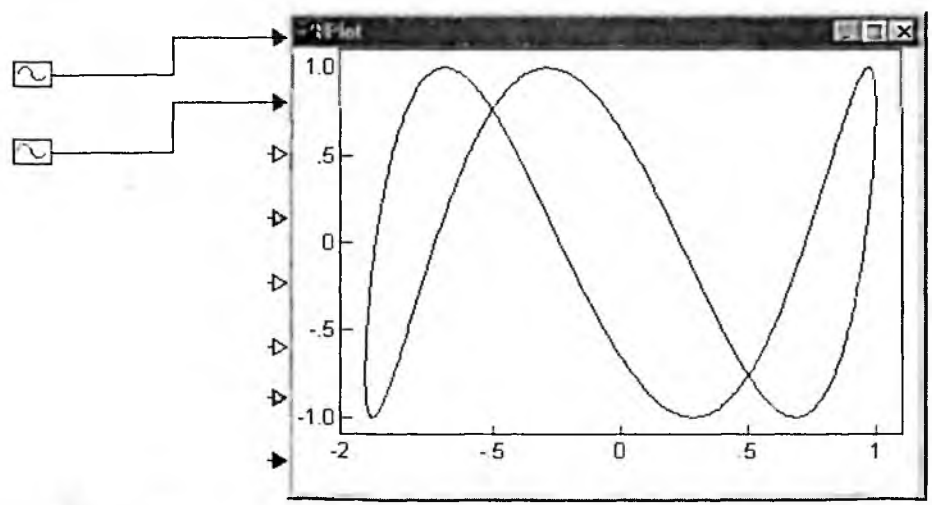


Рис. 3.79. Построенная фигура Лиссажу

Активизация опции Multiply XY trace позволяет задействовать до четырех независимых фазовых разверток осциллографа. Если же опция не активна, то развертка по горизонтальному каналу будет осуществляться одним верхним входным сигналом для семи каналов вертикального отклонения. Пример построения трех фигур Лиссажу показан на рис. 3.80.

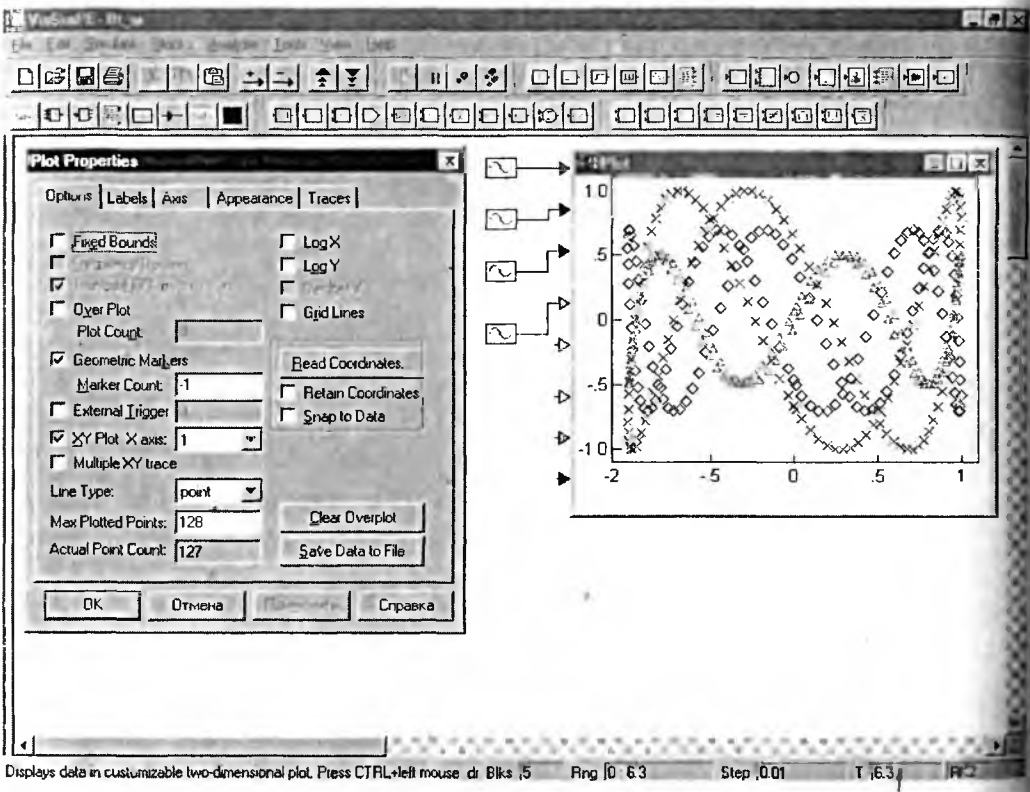


Рис. 3.80. Построение трех фигур Лиссажу по их точкам (меткам) и окно установки свойств осциллографа для этого случая

Возможность отображения множества кривых может использоваться для анимации изображений, например для показа изменений фазовых портретов в ходе моделирования систем и устройств. Заметим, что фазовые портреты требуют наличия у их регистратора двух входов — на один вход подается сигнал, а на другой его производная. Особенно часто фазовые портреты используются при анализе колебательных систем, описываемых системами или одиночными дифференциальными уравнениями второго порядка.

3.12.11. Блок остановки моделирования stop

Блок stop анализирует поданный на него сигнал и в зависимости от величины входного сигнала завершает процесс симуляции и может запретить авторестарт. Его действие отвечает следующим условиям:

- завершение моделирования с запретом авторестарта, если x > 2;

- завершение моделирования без запрета авторестарта, если x > 1;
- моделирование в обычном режиме, иначе.

Завершая текущую сессию моделирования досрочно (рис. 3.81), можно заметно уменьшить время моделирования. Наибольший эффект от использования блока можно получить в случае, если модель требует серии повторных прогонок, например когда активизирован режим авторестарта.

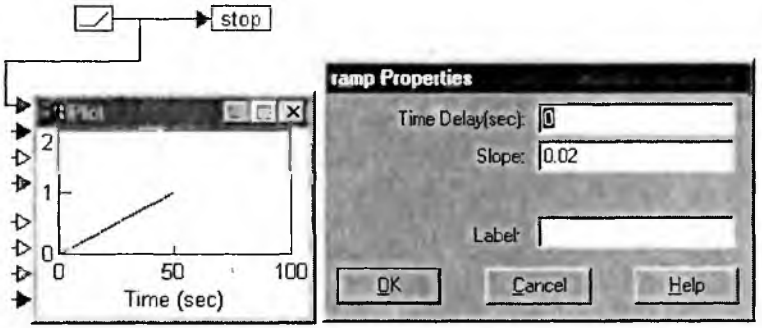


Рис. 3.81. Пример применения блока stop (моделирование завершается при достижении сигналом на входе блока stop значения 1)

Предопределенная переменная программы VisSim $runCount хранит номер текущей симуляции в режиме авторестарта и может быть использована для изменения параметров или переключения структуры блок-схемы.

3.12.12. Блок самописца stripChart

Блок stripChart является универсальным виртуальным прибором для визуализации долго протекающих процессов. Его аналогом является ленточный самописец. Блок самописца во многом напоминает осциллограф, но он отличается от него применением более узкого временного окна просмотра и наличием снизу линейки прокрутки, с помощью которой можно перемешать окно просмотра по записи сигнала (рис. 3.82).

Окно свойств самописца также очень напоминает окно свойств осциллографа. Отличия видны в основном на вкладке Options (рис. 3.83). Эта вкладка содержит почти те же установки, что и у осциллографа, но их заметно мень-

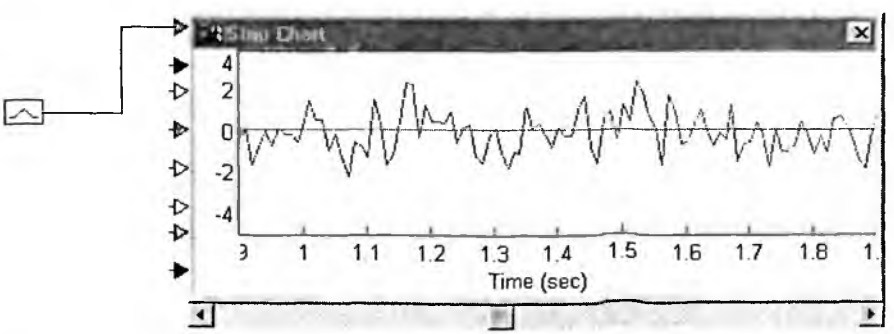


Рис. 3.82. Просмотр шумового сигнала от генератора Гауссового шума с помощью самописца

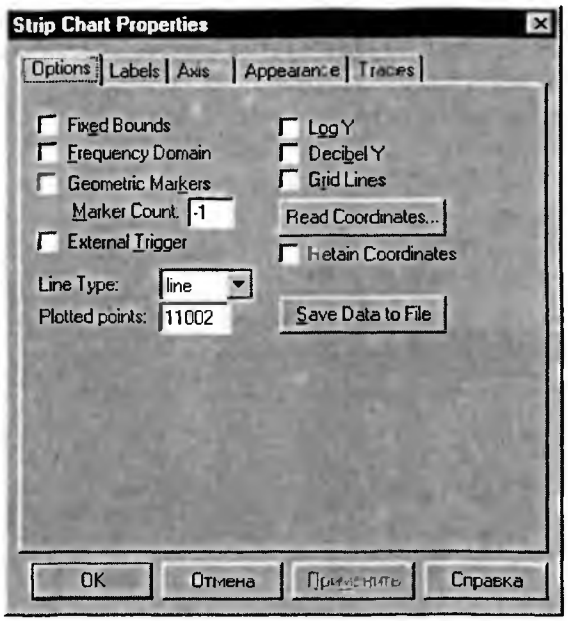


Рис. 3.83. Окно свойств самописца с открытой вкладной Options

ше. Остальные вкладки практически аналогичны описанным выше для окна осциллографа.

Еще один наглядный пример применения самописца и осциллографа приведен на рис. 3.84. Здесь с помощью трех генераторов синусоидальных ко-

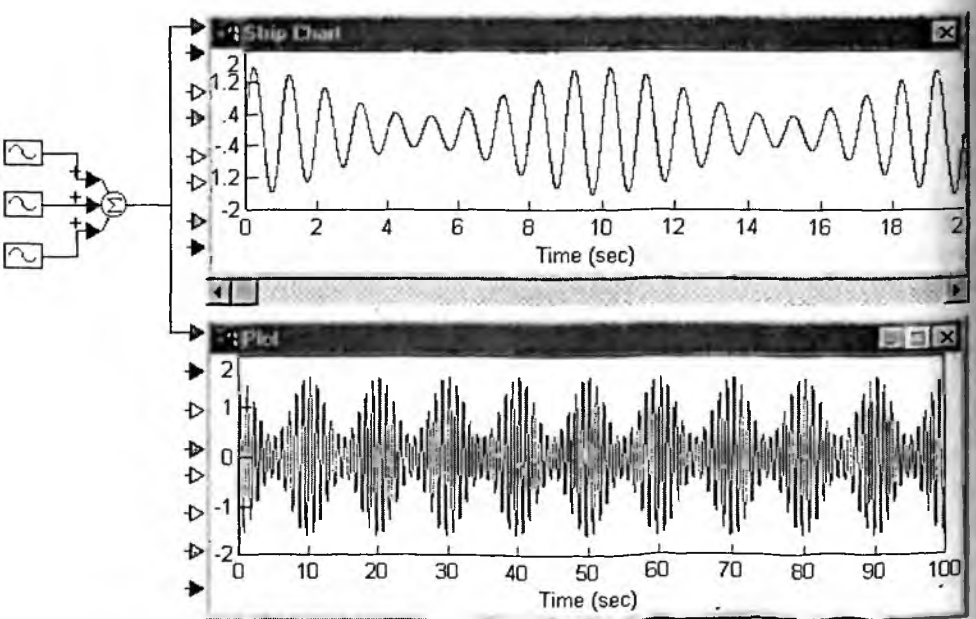


Рис. 3.84. Просмотр амплитудно-модулированного колебания самописцем (сверху) и осциллографом (снизу)

лебаний синтезирован сигнал, характерный для амплитудной модуляции. Средний генератор задает несущую частоту, а два других — частоты, увеличенные и уменьшенные на более низкую частоту модуляции. Нетрудно заметить, что в окне самописца сигнал виден лучше, поскольку окно выделяет лишь часть общего интервала времени моделирования.

В окне самописца можно проводить и спектральный анализ представленного в нем сигнала. Пример этого дан на рис. 3.85. При использовании логарифмического масштаба по вертикальной оси и ограничении диапазона частот можно наблюдать не только общий спектр (кривая сверху), но и три частоты спектра амплитудно-модулированного колебания (три пика на кривой спектра снизу).

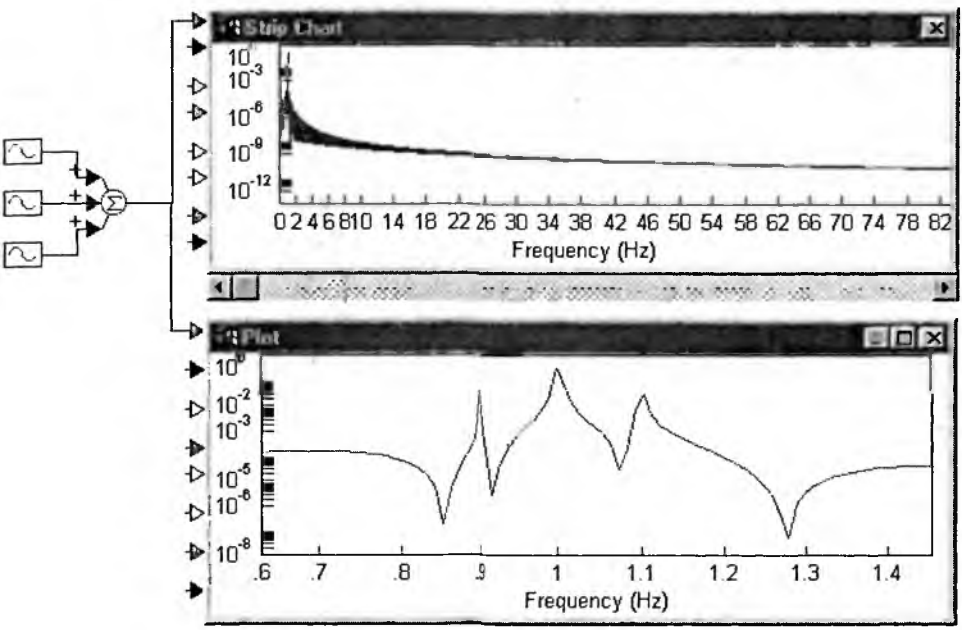


Рис. 3.85. Построение спектра прямо в окне самописца

К сожалению, надо отметить, что спектры в окне осциллографа и в окне самописца достаточно далеки от теоретических и скорее напоминают спектрограммы приборов — спектроанализаторов. Например, спектр амплитудно-модулированного колебания теоретически представляется всего тремя вертикальными линиями — несущего колебания и боковых частот.

3.13. Блоки генераторов сигналов

3.13.1. Блок «кнопка» button

Блок «кнопка» button служит для задания кнопки, реагирующей на нажатие мышью. Этот блок изначально имеет вид прямоугольника с надписью button, но можно придать ему вид кнопки выбором изображения из BMP-файла.

Созданная кнопка при активизации мышью может возвращать от 2 до 16 состояний (дискретов). Возвращаемый сигнал y = state – 1.

Окно свойств этого блока открывается при нажатии вначале клавиши Ctrl с удержанием, а затем уже правой клавиши мыши. Вид окна показан на рис. 3.86.

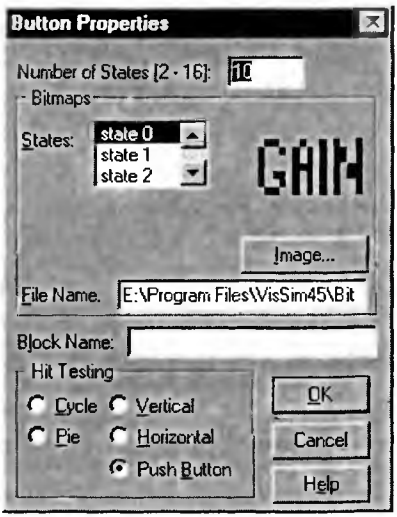


Рис. 3.86. Окно свойств блока button

Для каждого состояния можно выбрать свое изображение кнопки. После выбора изображения для состояния (из прямо указываемого файла или файла, найденного его поиском) изображение замещает слово «button». Можно также задать текстовое имя кнопки и определить тип ее функционирования:

- Cycle — циклическое изменение выходного сигнала с шагом +1;
- Pie — выбор состояния из круговой диаграммы;
- Vertical — выбор состояния из столбца;
- Horizontal — выбор состояния из горизонтальной линейки;
- Push Button — выбор состояния нажатием на кнопку.

В приведенных в данной книге блок-схемах моделей этот блок не используется.

3.13.2. Блок задания константы const

Блок const мы уже неоднократно применяли для создания численных констант. В общем случае он генерирует либо постоянный сигнал, либо матрицу элементов-констант, либо алфавитно-цифровую текстовую строку. Итак, блок может выполнять следующие функции:

y = const y = [constij] y = "текст"

Диалоговое окно свойств блока позволяет задать генерируемый сигнал (значение по умолчанию 1). Для генерации постоянного сигнала достаточно

ввести его значение. Для генерации матрицы элементов-констант в окне свойств в строке Value надо ввести требуемую совокупность числовых значений, следуя правилам:

- всю совокупность вводимых числовых значений заключите в квадратные скобки;
- элементы матрицы отделите пробелами или запятыми;
- строки матрицы отделите точкой с запятой.

Для генерации алфавитно-цифровой текстовой строки в строке ввода Value введите требуемый текст, заключив его в кавычки. Примеры задания констант показаны на рис. 3.87.

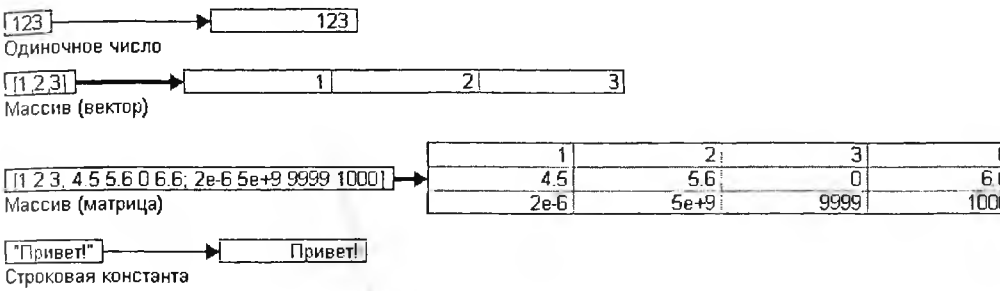


Рис. 3.87. Примеры задания констант разного типа

3.13.3. Блок задания диалоговой константы dialogConstant

Блок dialogConstant генерирует постоянную величину (константу) y = const, которая должна являться либо параметром, либо начальным условием. Блок предназначен для облегчения процесса реконфигурации блок-схемы, скрытой в составном блоке, без необходимости перехода внутрь ее. С его помощью создаются диалоговые окна составных блоков.

Под реконфигурацией понимается изменение параметров и начальных условий блок-схемы. Непосредственно изменить начальные условия на интеграторах с помощью блоков dialogConstant нельзя, так как нельзя подать сигнал в диалоговое окно. Эта операция выполняется внешним способом с применением сумматора (на входы сумматора подаются сигналы с интегратора и блока dialogConstant).

3.13.4. Блок импорта данных import

Блок импорта данных генерирует сигнал или синхронную совокупность сигналов, считывая выборки из .dat, .m, .mat или .wav-файла. Генерация сигнала может быть организована либо с жестким интервалом, либо согласно временной метке, прописанной для каждой выборки в одной из колонок файла (псевдоасинхронный источник сигнала). Применение блока импорта данных для приема данных из файла, созданного блоком экспорта данных export, было показано на рис. 3.59.

3.13.5. Блок генерации параболического сигнала parabola

Блок parabola генерирует сигнал, изменяющийся с постоянным ускорением:

$$y = a \cdot (t - t)2,$$

где 2a — ускорение; t — временное смещение; t — текущее время симуляции (не путать с реальным). Пример применения блока показан на рис. 3.88.

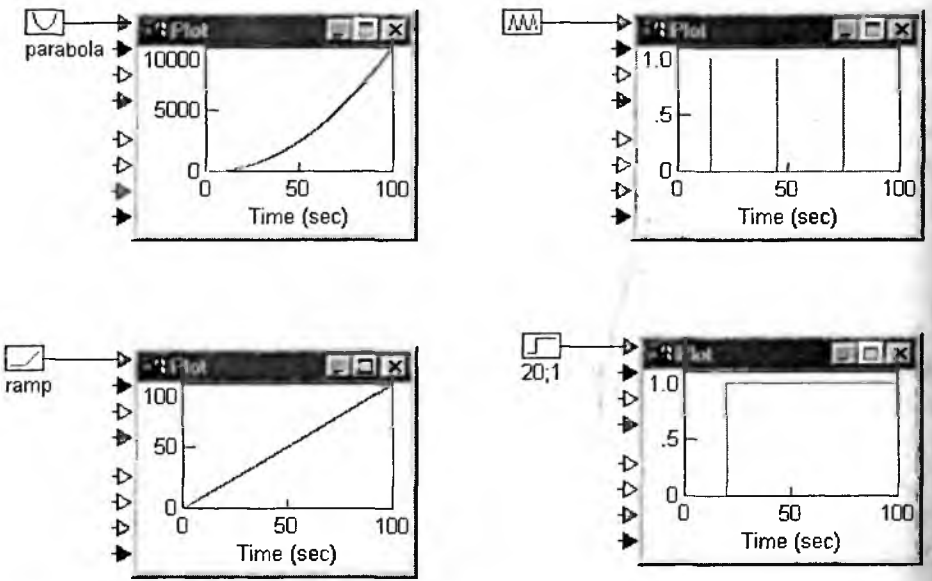


Рис. 3.88. Примеры применения блока parabola, pulseTrain, ramp и step

В окне свойств блока устанавливается время задержки Time Delay, параметр Slope rate (Ускорение/2) и метка Label.

3.13.6. Блок генератора запускающих импульсов pulseTrain

Блок генератора запускающих (синхронизирующих) импульсов генерирует короткие импульсы единичной амплитуды, которые используются для синхронизации других блоков, например регистра задержки, осциллографа и др. Импульсы соответствуют уравнению:

$$y = d0[nT - t],$$

где d0[nT] — единичная импульсная решетчатая функция, n — номер дискретного момента времени, T — интервал следования импульсов, t — время запуска генератора (чистое запаздывание). Значение y = 1, если время моделирования совпадает с моментами времени nT, и y = 0, если иначе. Можно добавить блоку один или два входа для динамического изменения времени запуска генератора t = x1 и интервала следования импульсов T = x2. При этом статические параметры в диалоговом окне свойств блока будут перекрыты.

В диалоговом окне задается время задержки Time Delay, время между импульсами Time Between Pulses (период повторения импульсов, по умолчанию равный 0.01) и метка блока Label. Пример применения блока дан на рис. 3.88.

3.13.7. Блок генератора линейно-изменяющегося сигнала ramp

Блок ramp генерирует сигнал, изменяющийся с постоянной скоростью:

$$y = v(t - td),$$

где v — скорость (крутизна изменения сигнала), t — текущее время моделирования (его не надо путать с реальным временем), td — время временной задержки. В окне свойств блока можно установить время Time Delay (td), крутизну изменения сигнала Slope и метку блока. Пример применения блока ramp показан на рис. 3.88.

3.13.8. Блок генератора ступеньки step

Блок step генерирует единичную ступенчатую функцию, которая часто используется в качестве возмущающего воздействия для получения переходной функции системы — h(t). Работа блока соответствует выражению

$$y = Am \cdot 1(t–td),$$

где $1(t - td)$ — единичная ступенчатая функция, t — текущее время моделирования (не путать с реальным временем), td — время временной задержки ступени; Am — амплитуда ступени. Можно также описать работу блока step выражением

$y = 0$, если $t < t$,

$y = 1 \cdot Am$, если иначе.

В окне свойств задаются время задержки Time Delay, амплитуда Amplitude (Am) и метка блока (Label). Пример применения блока step дан на рис. 3.88.

3.13.9. Блок выдачи реального времени realTime

Блок realTime считывает текущее время с системных часов компьютера и генерирует сигнал, значение которого равно времени в миллисекундах с момента запуска процесса симуляции. Используя блок, вы можете оценить быстродействие фрагментов вашей блок-схемы. Пример применения блока представлен на рис. 3.89.

3.13.10. Блок генератора синусоидального сигнала sinusoid

Блок синусоидального сигнала sinusoid служит для создания синусоидального сигнала. Применение этого блока и его настройки мы уже неоднократно обсуждали. Блок выглядит как прямоугольник с изображением одного перио-

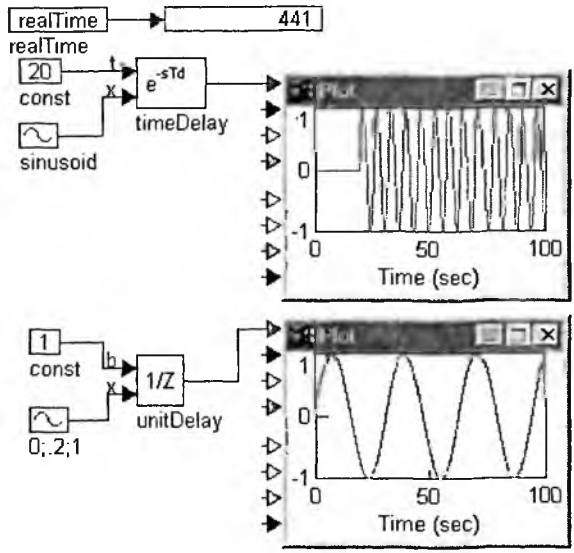


Рис. 3.89. Примеры применения блоков realTime, sinusoid и блоков задержки

да синусоиды. В окне свойств блока можно установить временную задержку (фазу), амплитуду и частоту синусоиды.

Блок sinusoid генерирует сигнал синусоидальной формы:

$$y = Am \sin(w \cdot (t - td)),$$

где Am — амплитуда; w — круговая частота ($2\pi f$); t — текущее время симуляции (не путать с реальным); td — временное смещение. Меняя параметр td, можно произвольно менять начальную фазу сигнала.

В окне свойств блока можно задать время задержки Time Delay (td), частоту Frequency [Hz либо Rad/Sec], амплитуду Amplitude (Am) и метку блока Label. По умолчанию частота генерируемого синусоидального сигнала равна 1 [рад/с].

3.13.11. Блок регулируемого постоянного сигнала slider

Блок slider генерирует постоянный сигнал, величину которого можно плавно менять, перемещая движок регулятора (слайдера) с помощью мыши. Это очень напоминает регулировку напряжения реостатом или потенциометром. Для более точного позиционирования движка мышью можно увеличить размер блока активацией опции High Precision Display в диалоговом окне, вызываемом командой меню Edit > Preferences.

Для изменения сигнала с произвольным приращением перемещайте мышью движок регулятора. Для изменения сигнала на фиксированное приращение щелкайте мышью по оси движка. Для активации программного (автоматического) изменения сигнала с фиксированным приращением нажмите левую кнопку мыши в требуемой конечной позиции на оси движка и, сместив указатель за пределы блока, отпустите кнопку (можно просто ее удерживать

на оси требуемое время). Для установки прецизионного значения откройте диалоговое окно свойств.

Примеры применения блока slider представлены на рис. 3.90. Там же показано окно свойств данного блока.

В окне свойств блока slider можно задать следующие установки: Current Value (Текущее значение, по умолчанию равное 0), Upper Bound (Верхний предел m, по умолчанию равный 100), Lower Bound (Нижний предел, по умолчанию равный –100), Increment (Приращение, по умолчанию 1) и Label (Метка блока). Приращение можно задать либо в абсолютных, либо в относительных единицах, установив в требуемое положение опцию «%».

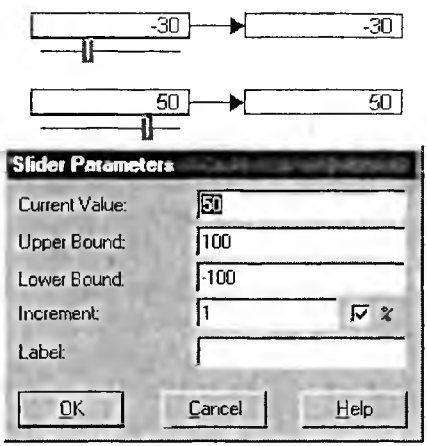


Рис. 3.90. Примеры применения блока slider

3.14. Блоки задержки

3.14.1. Блок временной задержки timeDelay

Блок timeDelay служит для задания чистого запаздывания для входного сигнала. Его можно динамически менять в процессе моделирования. Для этого блок имеет вход t (верхний). Блок построен на буфере элементов памяти по принципу первый вошел — первый вышел. На первом шаге симуляции буфер заполняется значением, равным начальному условию. Сигнальный вход *x* расположен ниже.

Блок реализует функцию

$$y = x(t - td),$$

где t — текущее время моделирования (не путать с реальным); td — величина чистого запаздывания, или

$y = 0$, если $t < td$,
$y = x(t - td)$, если иначе.

В окне свойств блока задаются: Initial Condition (Начальное условие, определяющее выходной сигнал на промежутке времени $0 < t < td$ и по умолчанию равно 0), Max Buffer Size (Размер буфера, задает количество элементов памяти в буфере блока, по умолчанию 4000) и Label (Метка блока). Пример применения блока дан на рис. 3.89.

При наличии блока timeDelay во фрагменте блок-схемы VisSim не может определить ее частотную характеристику с помощью своей библиотеки анализа. Для решения этой задачи следует построить аппроксиматор звена чистого запаздывания либо на совокупности апериодических звеньев первого порядка, либо на фазосдвигающих звеньях.

3.14.2. Блок регистра задержки unitDelay

Блок unitDelay задерживает выборки сигнала на одну дискрету времени, период которой определяет внешняя синхронизирующая последовательность (внешний синхросигнал). Блок unitDelay при моделировании дискретных систем играет ту же основополагающую роль, что и блок, интегратор при моделировании непрерывных систем.

Между синхроимпульсами выходное значение поддерживается неизменным с помощью экстраполятора нулевого порядка. Синхровход обозначен буквой *b* (от boolean — логический). Сигнальный вход — буквой *x*. Блок активизируется на текущем шаге симуляции, если на логическом входе активный уровень 1. При этом сначала хранимые регистром данные передаются экстраполятору (до следующего синхроимпульса), а только после этого содержимое регистра обновляется входным сигналом, чем и достигается эффект задержки. Пример применения блока дан на рис. 3.89.

В окне свойств блока устанавливаются следующие параметры: Initial Condition (Начальное условие, определяющее значение выходного сигнала до момента подачи синхроимпульсов и равное по по умолчанию 0), ID (Не использовано и зарезервировано на будущее), Checkpoint State (Сохраненное состояние) и Label (Метка блока).

3.15. Средства анимации

3.15.1. Условия наблюдения анимационных графиков

Анимация — это средства создания «живых» изображений. Обычно анимация реализует принцип, применяемый при создании мультипликационных фильмов. Он заключается в показе ряда меняющихся файлов — фреймов. Библиотека блоков VisSim имеет два анимационных блока: блок Animation для покадрового показа ряда изображений, представленных bmp-файлами, и блок анимации отрезка прямой lineDraw путем пошагового изменения координат его начала и конца.

Анимацию в VisSim можно наблюдать только в том случае, если активирован режим Display. В этом режиме (он оказывает действие только на текущий уровень блок-схемы) видима только та часть блоков, которая может быть использована для оформления этого режима. Соединения и составные блоки (по указанию) скрываются. Нельзя также редактировать блок-схему, перемещать блоки, менять параметры. Но остается возможность вносить коррекцию в процесс симуляции модели посредствам управляющих элементов — блоков ползунковых регуляторов, кнопок и констант. Кроме того, переход на другой уровень блок-схемы (для которого не активирован режим) позволяет вносить любые изменения. После выполнения и просмотра анимации можно обновить вид рабочего поля командой Repaint Screen в позиции Edit меню.

3.15.2. Блок анимации линии lineDraw

Блок анимации линии lineDraw хорошо подходит для визуализации движения рычажных механизмов, маятников и других устройств. Этот блок имеет 4 входа для задания координат *x* и *y* начала и конца отрезка прямой. Если эти координаты меняются, то меняется положение отрезка прямой. Рисунок 3.91 показывает простой пример применения этого блока. Сверху показано окно установки параметров моделирования с параметрами, обеспечивающими плавное перемещение отрезка прямой.

Верхняя точка отрезка в модели рис. 3.91 имеет фиксированные координаты (100, 100). Нижняя точка имеет фиксированную координату y = 300, тогда как координата x = 100 + 75*sin(t). Последнее и создают иллюзию вращательных колебаний (см. рис. 3.92), полученную при использовании режима Display. Заметим, что в этом примере длина отрезка несколько меняется в процессе колебаний.

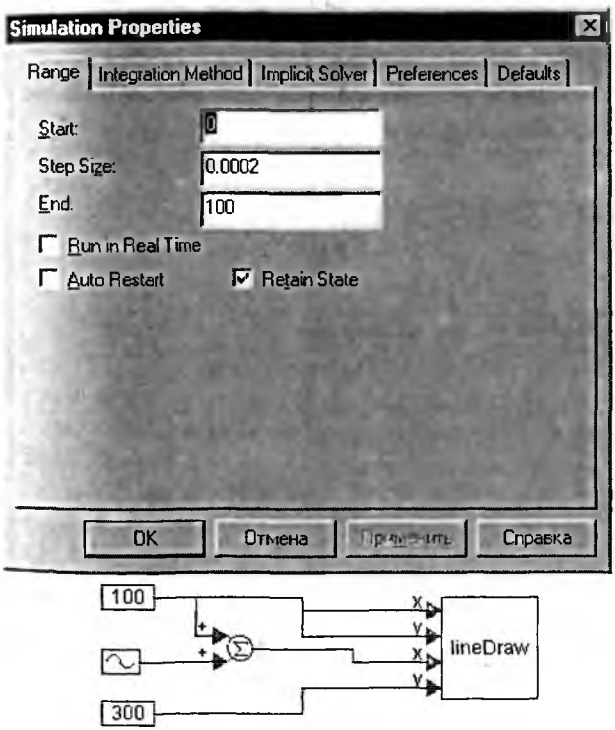


Рис. 3.91. Модель для анимации (вращательных колебаний) отрезка прямой

Рис. 3.92. Один из кадров анимации для модели рис. 3.91

Окно свойств блока lineDraw показано на рис. 3.93. Окно позволяет задать толщину линии отрезка прямой Thickness, стиль линии (сплошная, пунктирная, из точек и др.) и ее цвет. Стиль можно задать только при единичной толщине линии, в ином случае список стилей не действует.

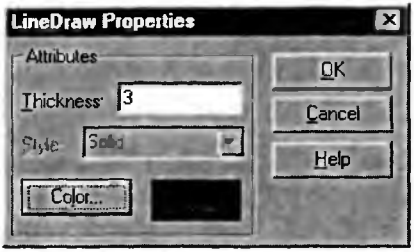


Рис. 3.93. Окно свойств блока lineDraw

3.15.3. Блок анимации Animation

Для создания анимации объектов произвольного вида служит блок Animation. Его изображение и окно свойств показаны на рис. 3.94.

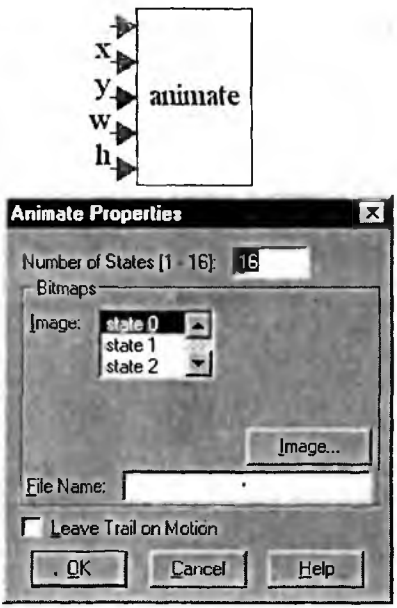


Рис. 3.94. Изображение блока анимации Animation и окно свойств этого блока

Для получения анимации произвольного объекта нужно создать набор кадров этого объекта в виде BMP-файлов растровой графики с количеством цветов не более 256. Создать набор таких файлов можно, например, с помощью даже простейшего графического редактора Paint.

С помощью окна свойств блока Animation в окошке Number of States можно задать число кадров анимации — от 1 до 16. В меню Image укажите «state0». Далее выполните любую последовательность действий.

- Нажмите кнопку Image для вызова системного диалогового окна выбора файла. Установив в диалоговом окне BMP-фильтр, выберите соответст

вующий номеру кадра BMP-файл. Его имя должно появиться в строке ввода File Name.

- Вручную введите путь и имя BMP-файла, соответствующего номеру кадра. Путь может быть абсолютным или относительным.
- Если требуется подключить второе изображение, то в списке выбора Image укажите «state1» и повторите описанные действия.
- Нажмите кнопку OK или клавишу ENTER.

Сигналы, поданные на входы блока анимация и управляют одноименным процессом в течение симуляции модели. Блок анимации имеет пять входов, назначение которых описано ниже.

Сигнал на верхнем входе определяет номер отображаемого кадра. При подключении к блоку кадры нумеруются числами 0, 1, ..., 15. Соответственно, значение входного сигнала должно быть округленным к наименьшему целому числу. Если значение входного сигнала выйдет за диапазон номеров подключенных кадров, то соответственно будет отображаться либо нулевой, либо имеющий наибольший порядковый номер кадр.

Сигналы, подаваемые на входы *x* и *y*, определяют координаты верхнего левого угла (в пикселях дисплея) для отображения текущего кадра на рабочем поле программы VisSim. Нулевым координатам соответствует верхний левый угол рабочего поля. Координаты увеличиваются к нижнему правому углу. Для отображения анимации на большинстве мониторов рекомендуется придерживаться видеорежима VGA (640×480).

Если подаются сигналы на входы w и h блока, то при анимации размеры текущего кадра будут ограничены по ширине и высоте соответственно. Тем самым возможно, например, создание эффекта приближения.

Опция Leave Trail on Motion (Оставлять след от движения) позволяет оставлять след от движущегося в ходе анимации объекта.

3.15.4. Другие средства анимации

Виртуальные блоки осциллографа, стрелочные, линейные и «светодиодные» индикаторы также обладают полезными возможностями в создании эффектов анимации. Осциллограф, к примеру, может применяться для демонстрации перемещения точки — нужно лишь намеренно замедлить моделирование.

Хорошее впечатление оставляет плавное перемещение стрелок стрелочных измерительных приборов и особенно работа линейных индикаторов. Мигание «светодиодных» индикаторов тоже выглядит живо и интересно. Примеры применения этих блоков представлены в модели, показанной на рис. 3.95. Работа этой модели вполне очевидна.

Пример применения осциллографа для простейшей анимации дан на рис. 3.96. Здесь имитируется работа бака, заполняемого водой (это и отображает верхний осциллограф). Когда уровень воды достигает критической отметки, срабатывает логика модели (блок Control Logic) и бак быстро опорожняется, после чего наступает его медленное заполнение. Это напоминает известную модель сливного бачка в туалете. Нижний осциллограф строит временную зависимость уровня воды в баке.

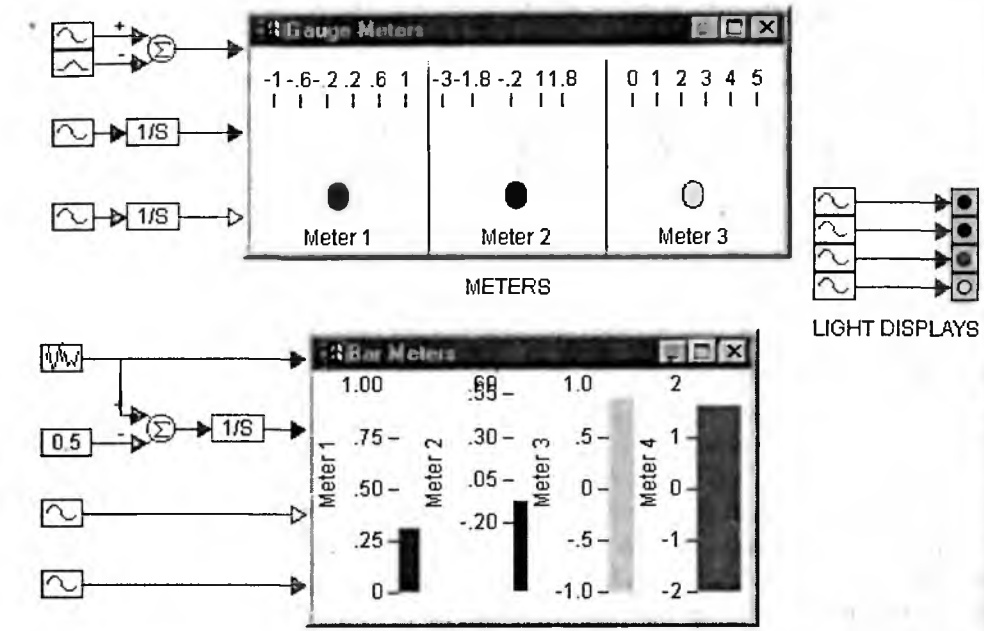


Рис. 3.95. Примеры применения «стрелочных» и линейных индикаторов, а также линейки светодиодов

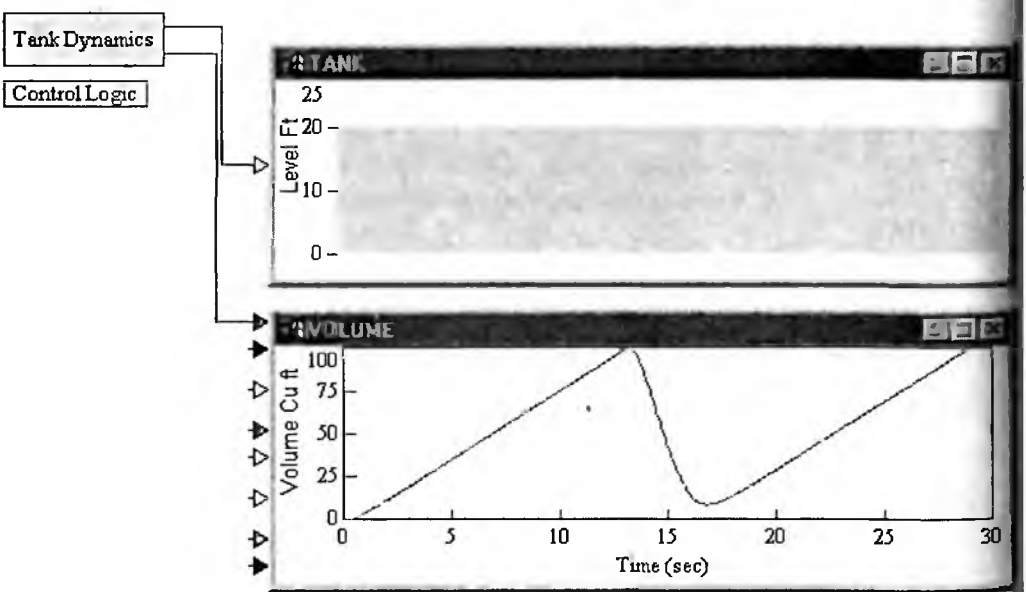


Рис. 3.96. Модель заполнения водой и опорожнения бака

По существу, это типичная модель релаксационного процесса. С ее деталями можно ознакомиться, активизируя блоки модели мышью. Модель содержится в файле lvlcntl.vsm.

3.16. Дополнительные возможности VisSim 5

3.16.1. Расширенные средства матричных операций

В новую версию VisSim 5 введены расширенные средства матричных операций. В разделе Matrix введен блок diag для задания диагональной матрицы. Элементы диагонали задаются входным вектором, который и определяет размер матрицы. Например, если вектор задается как значение константы [1, 2, 3], то будет создана матрица с размером 3×3, у которой диагональные элементы будут равны 1, 2 и 3, а остальные элементы — нули. Параметр diagonal offset (по умолчанию 0) задает сдвиг диагонали на целое число. Если оно положительное, сдвиг диагонали задается вверх, а если отрицательное, то вниз.

Расширен синтаксис задания констант. Теперь в круглых скобках можно задавать ранжированные переменные, используя следующий синтаксис:

(start:[step:]end),

где start — начальное значение первого элемента вектора; step — необязательный шаг и end — конечное значение последнего элемента вектора. Например, (1:5) создает вектор из чисел 1, 2, 3, 4 и 5, а (1: 0.5: 2) создает вектор с элементами 1, 1,5 и 2. Примеры применения матричных операций представлены на рис. 3.97.

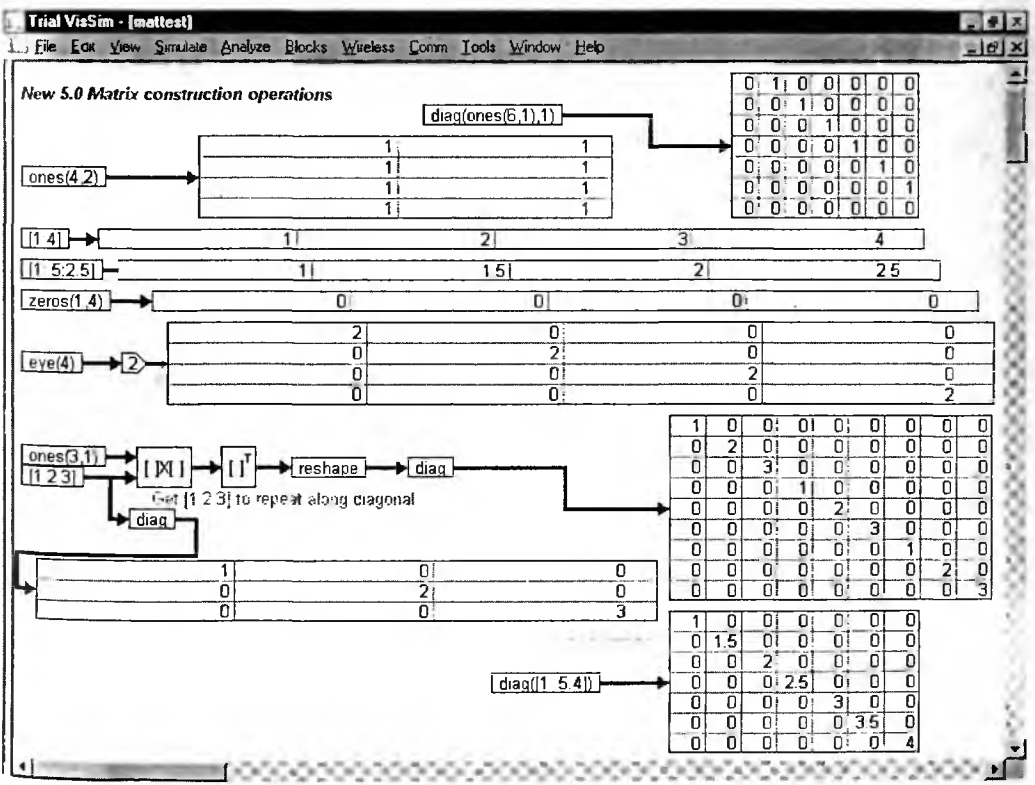


Рис. 3.97. Примеры матричных операций в VisSim 5

Расширенные матричные операции облегчают подготовку моделей, в которых широко используются матрицы. Но все эти операции чуть сложнее реализуются и в предшествующих версиях VisSim.

3.16.2. Расширенные операции с комплексными числами

В VisSim 5 расширены также операции с комплексными числами вида $Z = a + j \cdot b$, где a — действительная часть числа, b — мнимая часть числа и j — мнимая единица (квадратный корень из –1). Теперь при задании констант можно в круглых скобках задавать комплексные числа в виде (a, b). Если данные содержат такие числа, то отображающий их цифровой измеритель будет представлять как действительную, так и мнимую часть каждого данного. Примеры применения комплексных данных даны на рис. 3.98.

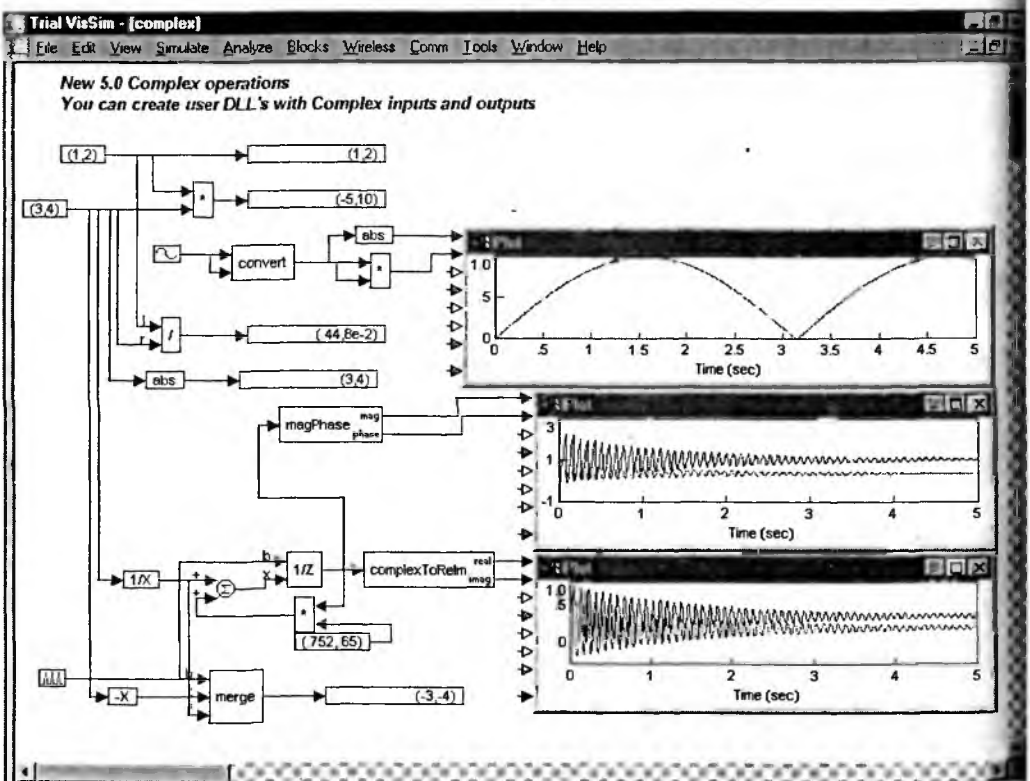


Рис. 3.98. Примеры работы с комплексными данными (числами)

Расширенные операции с комплексными числами облегчают моделирование тех систем, в которых широко используется аппарат комплексных чисел. Пожалуй, особенно широко он используется при анализе линейных электрических и радиотехнических цепей.

3.16.3. Новые блоки генерации сигналов

В VisSim 5 расширен и набор блоков для генерации импульсных сигналов. Новые блоки и их применение показаны на рис. 3.99. Здесь же показана возможность отображения многих кривых в окне виртуального осциллографа и окно свойств последнего.

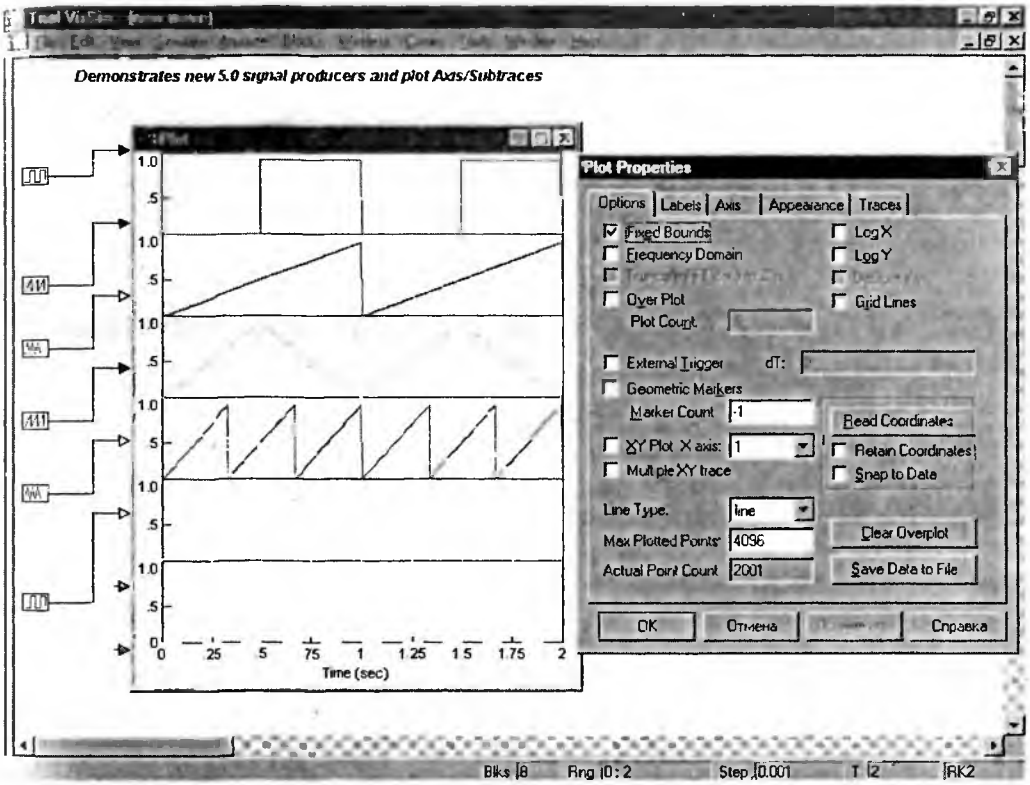


Рис. 3.99. Примеры применения новых блоков VisSim 5 для генерации импульсных сигналов

Конечно, новые блоки расширяют возможности синтеза сигналов, но все они легко реализуются и средствами стандартной библиотеки предшествующих версий VisSim. Альтернативой им является и генерация сигналов с помощью комбинаций элементарных функций.

Глава 4
Моделирование и математическая обработка сигналов

В этой главе мы познакомимся со средствами системы VisSim, предназначенными для моделирования и обработки сигналов, а также с многочисленными примерами, использующими эти средства. Эти средства не только решают многие самостоятельные задачи, но и являются частью средств моделирования более сложных систем и устройств, например фильтров различного типа.

4.1. Применение блоков из специализированных библиотек

VisSim 4.5 поставляется с рядом специализированных библиотек, в состав которых входят специальные блоки, которых нет в основной библиотеке (глава 3). Эти блоки предназначены для реализации некоторых сложных примеров, но их можно применять и в составе моделей пользователя. Кроме того, реализация блоков может служить хорошим примером для пользователей, создающих свои модели и не желающих «изобретать велосипед». В этой главе рассматриваются в основном блоки, ориентированные на решение таких массовых задач, как моделирование и обработка сигналов.

4.1.1. Блоки разделов Compnent

Множество дополнительных блоков расположено в папке Compnent. Этот раздел содержит следующие папки:

- Derivatv — блок вычисления производной;
- Dsp — примеры цифровой обработки сигналов (свертка, фильтр Калмана, вейвлет-преобразования);
- Dynsys — примеры моделирования динамических систем;
- Elecmech — примеры моделирования электромеханических систем;
- Electric — пример моделирования выпрямителя с конденсаторным фильтром;
- Hydrauli — модели блоков гидравлических устройств;
- Process — моделирование физических и химических процессов;
- Thermal — моделирование термических устройств.
- Turbine — моделирование системы управления турбиной.

В дальнейшем мы рассмотрим только те блоки и примеры из раздела Compnent, которые представляют достаточно широкий интерес. Заметим, что

самый большой раздел в этой библиотеке относится к блокам гидравлических устройств — они расположены в 12 папках. Но ввиду довольно узкой специфики их применения рассмотрение опущено.

4.1.2. Блоки раздела Toolbox

В разделе *инструментов* Toolbox имеются папки со следующим содержанием:

- Controls — множество блоков для построения систем контроля;
- Elecmech — блоки для моделирования электромеханических систем;
- Pade — блоки задания передаточных характеристик от первого до четвертого порядков;
- Siggen — блоки для задания наиболее распространенных сигналов;
- Tools — инструментальные блоки для определения параметров сигналов и осуществления широтно-импульсной модуляции.

Блоки этого раздела мы рассмотрим достаточно подробно, поскольку необходимость в их применения встречается повсеместно.

4.2. Блоки математической обработки

4.2.1. Блок дифференцирования непрерывных сигналов

В составе основной библиотеки VisSim нет блоков дифференцирования. Обычно для реализации этой важной операции используется блок интегрирования, так что *операция дифференцирования* рассматривается как операция, обратная операции интегрирования. Именно так реализован субмодуль аналогового дифференцирующего устройства deriv_a в составе набора инструментов Toolbox (папка Control) (рис. 4.1).

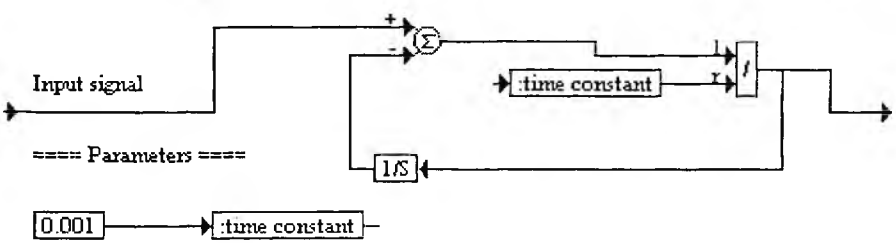


Рис. 4.1. Реализация блока аналогового дифференцирования

Единственным параметром блока является константа времени time constant, исходное значение которой равно 0.01. Блок реализует функцию

$$y(t) = \frac{d}{dt}x(t),$$

где $x(t)$ — входной сигнал, а $y(t)$ — выходной сигнал. Соответственно он имеет один вход и один выход. Как и некоторые другие блоки, реализованные на

уровне субблоков средствами основной библиотеки блоков, блок дифферен цирования имеет голубой фон. Вычисление производной осуществляется ко нечно-разностным методом за время time constant.

Пример применения этого блока представлен на рис. 4.2. На вход блок подается синусоида с амплитудой 2, вершины которой подрезаны на уровня –1,5 и 1,5 с помощью блока ограничения. Значение time constant взято ра ным 0,001, что обеспечивает достаточно высокое качество дифференцирова ния при временном шаге моделирования 0,01. Увеличение time constant за метно ухудшает качество дифференцирования. Полезно отметить выпадени точки с x = 0, в которой вычисленная производная равна 0, в то время как самом деле она конечна.

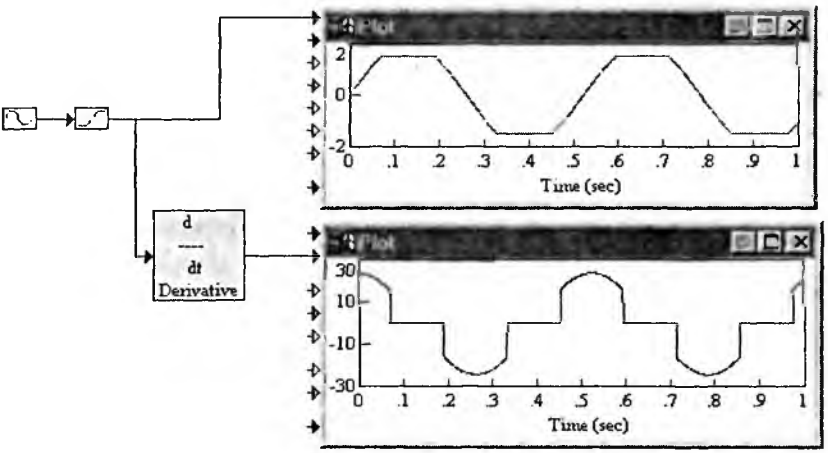


Рис. 4.2. Пример применения блока deriv_a

Следует отметить, что операция дифференцирования в отличие от опер ции интегрирования намного более чувствительна к импульсным помехам шумам в составе входного сигнала. Поэтому при моделировании надо прим нять эту операцию достаточно осторожно.

4.2.2. Блок дифференцирования дискретных сигналов

Блок *конечно-разностного вычисления производной* для дискретных систе deriv_d представлен диаграммой, показанной на рис. 4.3. Настраиваемым па раметром блока является Digital Update Time.

Рисунок 4.4 показывает пример применения блока deriv_d при значени Digital Update Time, равном 0,5. Нетрудно заметить, что в начале моделиров ния вычисленная производная резко отличается от ее идеального значени (производная от функции синуса есть косинус).

Описанные блоки фактически реализуют операцию *дифференцирования с сглаживанием*. Вследствие этого они не чувствительны к быстрым изменения входного сигнала, в частности к его дискретизации, связанной с моделиров нием при малом шаге интегрирования. Это может быть достоинством при р

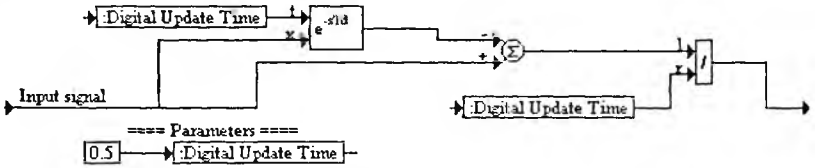


Рис. 4.3. Блок deriv_d для вычисления производных в дискретных системах

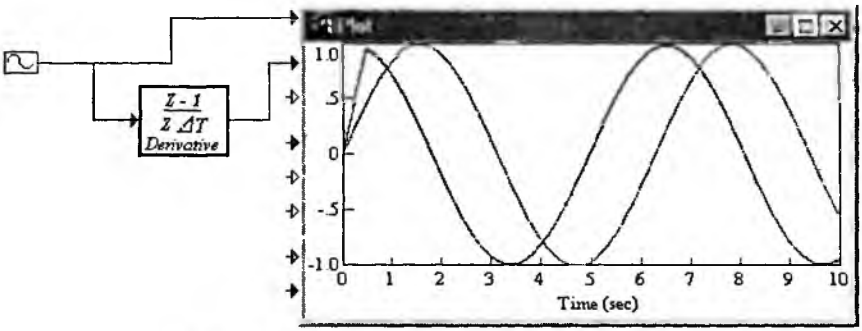


Рис. 4.4. Пример применения блока deriv_d

боте с зашумленными сигналами, но все же заметно снижает точность дифференцирования при быстрых изменениях сигнала. Поэтому представляет интерес блок Derivative из набора компонентов. Этот блок неделим и имеет два входа — верхний step-size и нижний signal для сигнала. Применение блока при значении параметра step size, равном 0,1, дано на рис. 4.5. Нетрудно заметить, что при этом блок отслеживает даже быстрые изменения сигнала, вызванные его дискретностью (наличием ступенек).

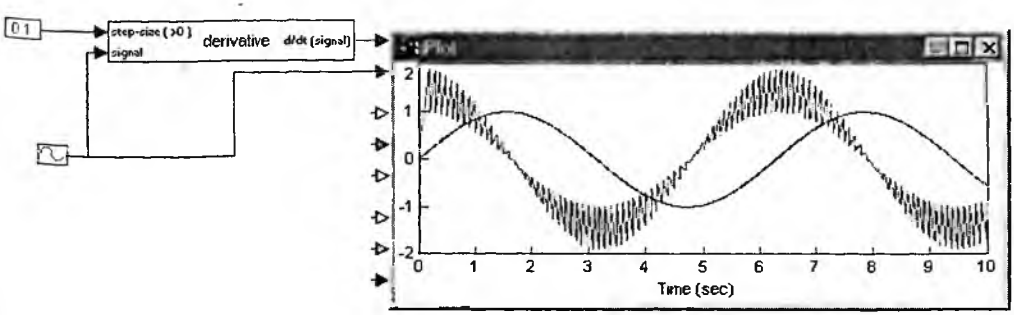


Рис. 4.5. Применение блока Derivative из набора компонентов при step-size = 0,1

Если уменьшить step-size до значения 0,01, равного шагу интегрирования при моделировании, то быстрые изменения входного сигнала, связанные с дискретностью, уже перестают влиять на работу блока. Этот случай показан на рис. 4.6.

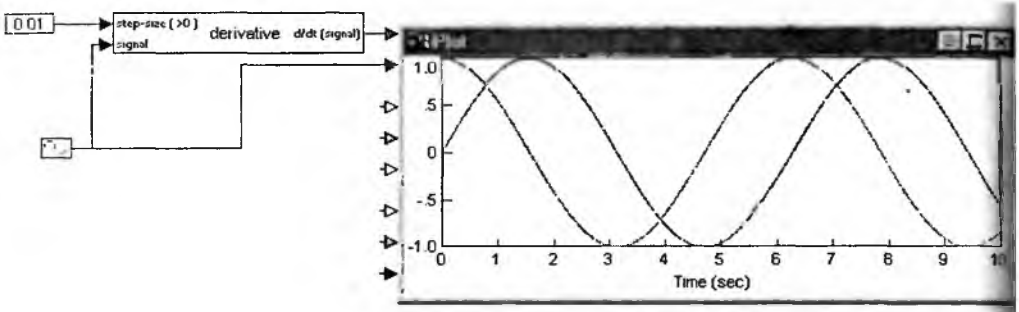


Рис. 4.6. Применение блока Derivative из набора компонентов при step-size = 0,01

4.2.3. Блок ограничения скорости слежения

У некоторых систем нелинейность проявляется в ограничении скорости слежения за изменяющимся сигналом. Часто это вызвано ограничением значений производной. Блок Rate Limiter моделирует это явление, что и показано на примере, представленном на рис. 4.7.

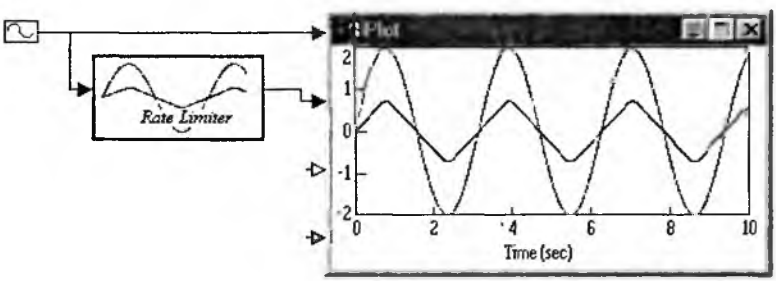


Рис. 4.7. Пример применения блока Rate Limiter, иллюстрирующий срыв слежения за синусоидальным сигналом

Диаграмма модели блока Rate Limiter представлена на рис. 4.8. Блок состоит их дифференцирующего блока (подобного блоку, представленному на рис 4.1) и интегратора с двухсторонним ограничителем между ними. Если значение производной входного сигнала не выходит за пределы ограничения, то выходной сигнал просто повторяет входной. В противном случае изменения сигнала происходят по линейно нарастающему или линейно падающему закону, что хорошо видно на рис. 4.8.

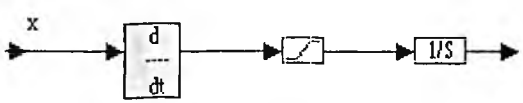


Рис. 4.8. Диаграмма блока Rate Limiter

4.2.4. Интегратор с автоматической инициализацией

В состав блоков раздела Control включен «интеллектуальный» интегратор, у которого автоматически вычисляется, оптимизируется и устанавливается уровень сигнала инициализации (установки начальных условий). Структурная схема такого интегратора, представленного блоком Trimmed Integrator, показана на рис. 4.9.

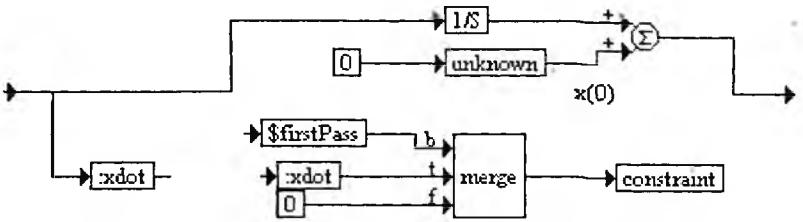


Рис. 4.9. Структурная схема блока Trimmed Integrator

Нередко применение такого интегратора проще, чем стандартного (из библиотеки), и может дать более высокую точность. Пример применения интегратора для интегрирования прямоугольных симметричных импульсов представлен на рис. 4.10. На выходе получаются треугольные импульсы, нижняя кромка которых привязана к нулю.

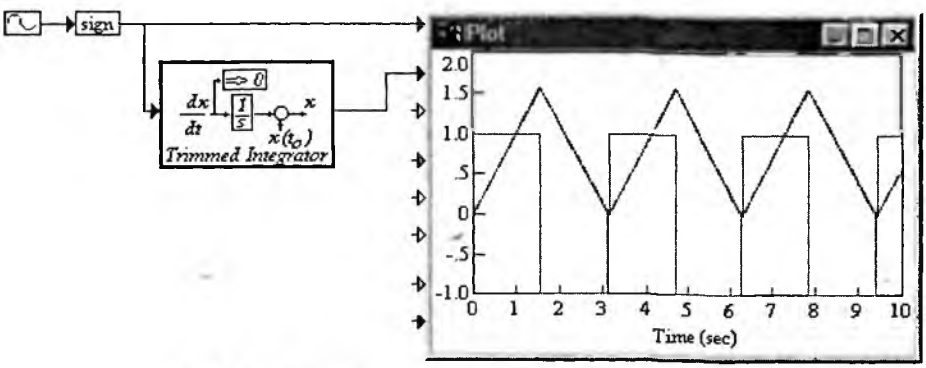


Рис. 4.10. Пример применения блока Trimmed Integrator

4.2.5. Блоки раздела Control для задания передаточных функций

Несколько блоков в разделе Control предназначены для задания передаточных функций конкретного вида для непрерывных и дискретных систем. Эти функции введены только для упрощения записи и реализации структурных диаграмм. Ввиду очевидности применения данных функций мы их рассматривать не будем. В диаграммах эти блоки выглядят как прямоугольники, в которых записаны соответствующие передаточные характеристики.

4.3. Блоки преобразования, контроля и генерации сигналов

4.3.1. Блоки преобразования аналоговых сигналов в квантованные и наоборот

В папке Elecmech набора модулей Toolbox есть два весьма полезных блок для преобразования аналоговых (непрерывных) сигналов в *квантованные* пр постоянном интервале времени между выборками и для обратного преобразо вания квантованных сигналов в аналоговые сигналы. Такие преобразовател широко применяются не только в электромеханических устройствах, но и в всевозможных системах обработки сигналов.

Первый вид преобразования обеспечивает блок a2d, диаграмма которо представлена на рис. 4.11. Работа блока задается тремя параметрами: число ступеней квантования (8, 12, 16 и т. д.) N, максимальным уровнем сигнал Max Input Value и шагом квантования по времени Digital Update Time.

Для обратного преобразования используется блок d2a. Его модель пре ставлена на рис. 4.12. Для настройки преобразователя используются парамет

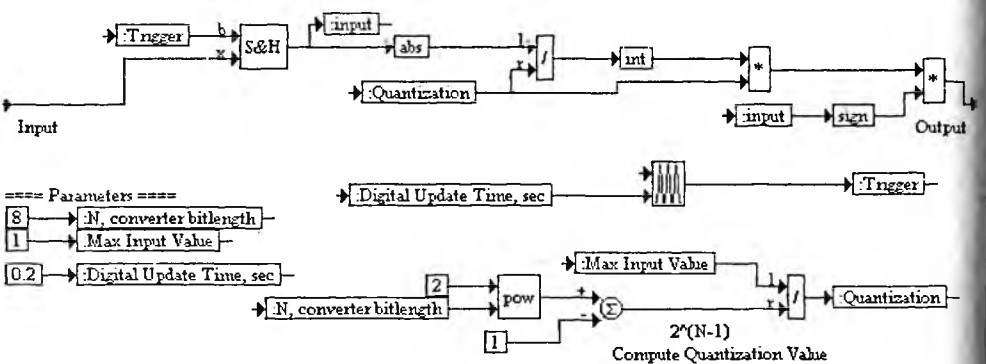


Рис. 4.11. Модель квантователя непрерывных сигналов

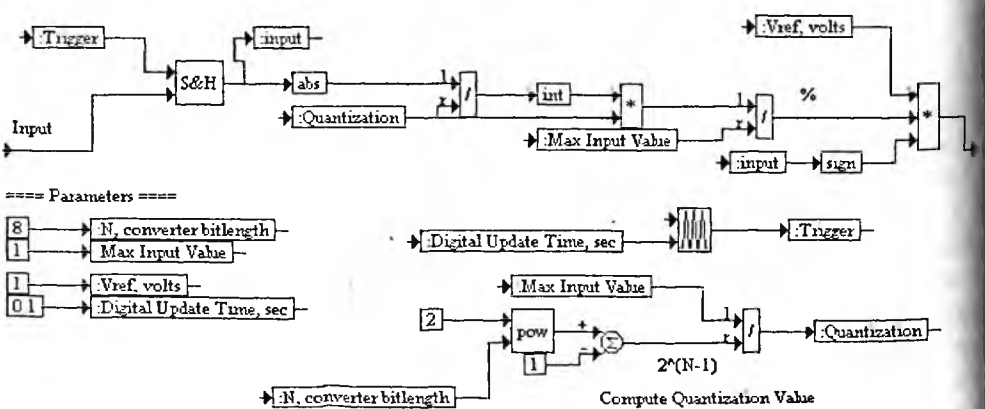


Рис. 4.12. Модель преобразователя квантованных сигналов в непрерывные

ры, уже указанные выше. Дополнительно задается уровень напряжения выходного сигнала Vref. Этот преобразователь все же дает на выходе ступеньки, но позволяет оценивать (интерполировать) уровень сигнала и в промежутках между ступеньками.

Рисунок 4.13 показывает применение этих блоков для обработки синусоидального сигнала. На выходе блока a2d получается ступенчатая кривая, вписывающаяся в синусоиду исходного сигнала. Как отмечалось, на выходе d2a полного восстановления сигнала все же не происходит. Однако даже с помощью простейшего фильтра удается существенно уменьшить погрешность восстановления, что обычно и реализуется на практике.

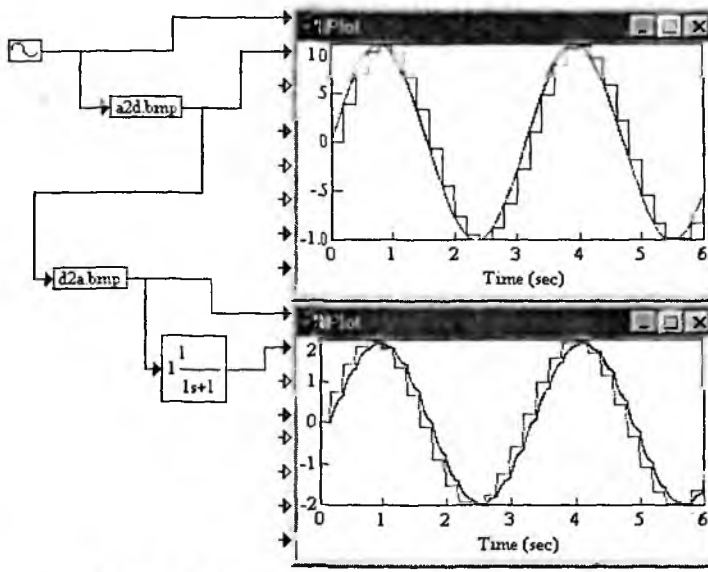


Рис. 4.13. Примеры применения блоков a2d и d2a

4.3.2. Блок канального мультиплексора

Еще один полезный блок из набора Elecmech — это блок *мультиплексора*, — устройства, по очереди подключающего входы к выходу. Модель четырехканального мультиплексора представлена на рис. 4.14. Число каналов мультиплексора можно при необходимости увеличить.

Рисунок 4.15 показывает подмодель одного из каналов мультиплексора. При увеличении числа каналов нужно увеличить число таких подмоделей — для мультиплексора рис. 4.14 таких субблоков применяется четыре.

Рисунок 4.16 показывает простой пример применения мультиплексора. На его входы поданы константы со значениями 1, 2, 3 и 4. В результате мультиплексор вырабатывает ступенчатое повторяющееся напряжение с уровнями ступенек, равными значениям уровней констант на входах.

Мультиплексор можно применять и для объединения нескольких изменяющихся во времени сигналов. На этом основано временное уплотнение сиг-

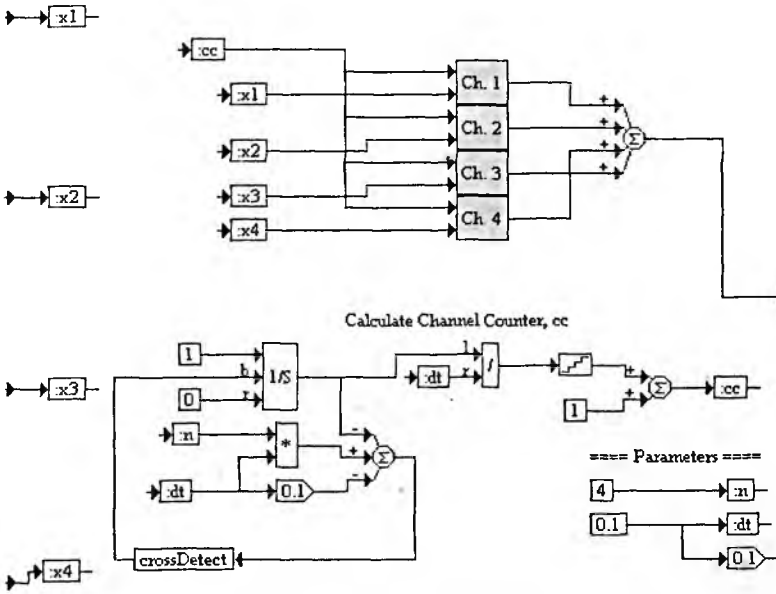


Рис. 4.14. Модель четырехканального мультиплексора

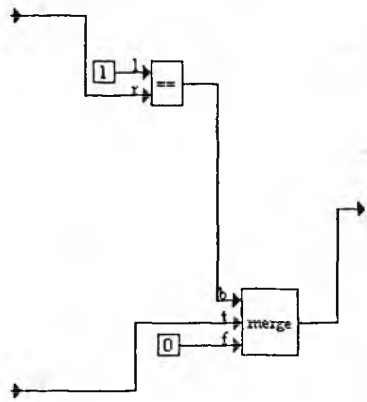


Рис. 4.15. Субблок одного из каналов мультиплексора

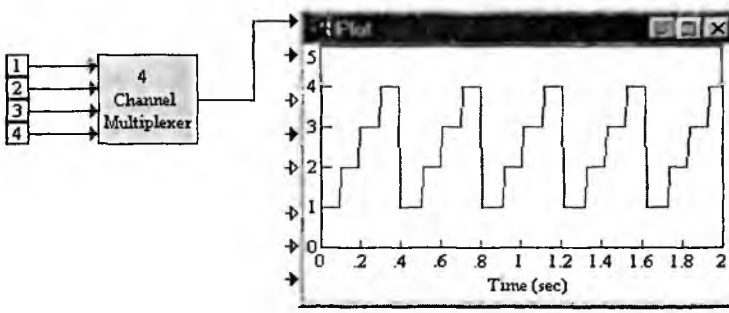


Рис. 4.16. Применение блока мультиплексора

налов. Разумеется, при этом предполагается, что за время передачи каждого сигнала он меняется незначительно.

4.3.3. Блок равномерного квантования по уровню

Помимо описанного квантования, равномерного по времени, широко используется квантование, равномерное по уровню. Блок Encoder реализует этот вид квантования. Его диаграмма представлена на рис. 4.17.

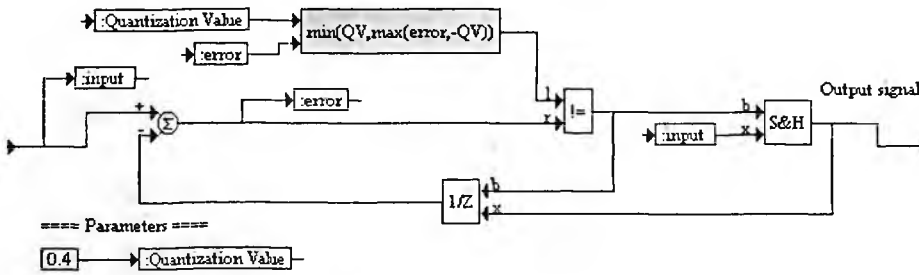


Рис. 4.17. Модель квантователя с равномерным квантованием по уровню

Этот вид квантования часто называют *дифференциально-кодовой модуляцией*, поскольку очередная ступенька квантованного сигнала появляется, как только разность между входным напряжением и предыдущей ступенькой не достигнет некоторого фиксированного уровня, который задается переменной Quantization Value (по умолчанию 0,4). В эту модель входит субблок min (QV, max (error, –QV)), диаграмма которого представлена на рис. 4.18.

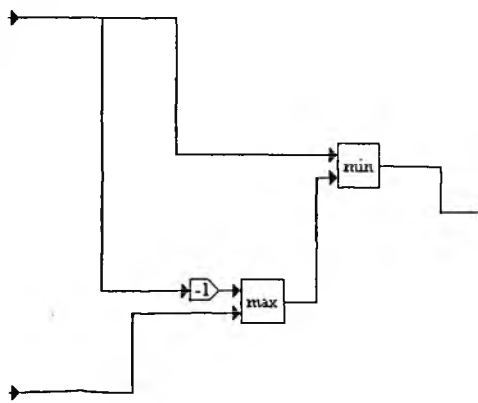


Рис. 4.18. Субблок min (QV, max (error, –QV))

Рисунок 4.19 показывает применение блока Encoder для квантования синусоидального напряжения. Естественно, что равномерность квантования по уровню ведет к его неравномерности по времени, и это хорошо видно из осциллограмм рис. 4.19.

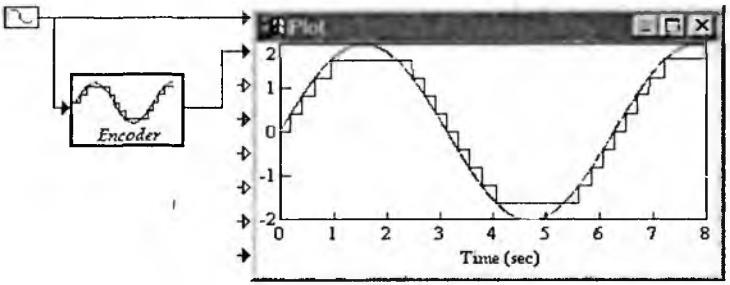


Рис. 4.19. Применение блока Encoder для квантования синусоидального напряжения

4.4. Блоки контроля сигналов из папки Tools

4.4.1. Блоки вычисления максимума и минимума сигнала

Одной из простейших и широко распространенных операций обработ сигналов является вычисление их *экстремумов* — *максимумов* и *минимумо* Модель вычислителя максимума значения сигнала представлена на рис. 4.2

В модуль входит субблок дискретной разностной модели, представленны на рис. 4.21. Он построен на блоке 1/Z и сумматоре.

Пример применения блока вычисления максимума синусоидального си нала представлен на рис. 4.22. Стоит обратить внимание, что блок вычисля максимум хотя и с малой, но заметной погрешностью (точное значение ма симума +1).

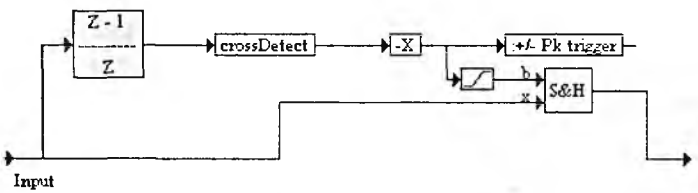


Рис. 4.20. Модель вычислителя максимума сигнала

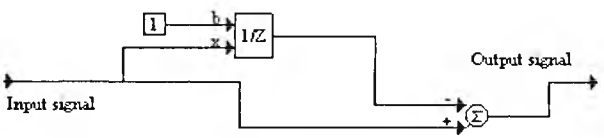


Рис. 4.21. Субблок дискретной разностной модели

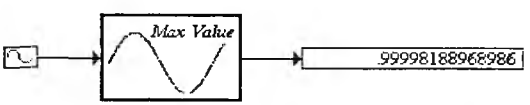


Рис. 4.22. Пример применения блока вычисления максимума сигнала

Аналогично реализован блок вычисления минимума сигнала (см. рис. 4.23). Он вычисляет значение минимума сигнала вне зависимости от знака минимуму. К примеру, у синусоиды значение минимума равно –1.

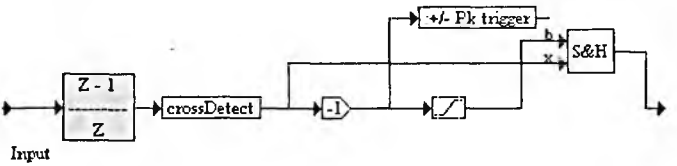


Рис. 4.23. Модель вычислителя минимума сигнала

Пример применения блока вычисления минимума синусоидального сигнала представлен на рис. 4.24. Как и в случае вычисления максимума сигнала, заметна погрешность вычисления минимума.

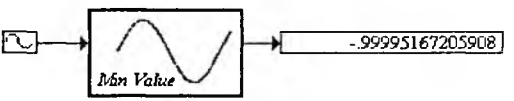


Рис. 4.24. Пример применения блока вычисления минимума сигнала

Следует помнить, что как максимум, так и минимум сигнала вычисляются за время моделирования. Если оно меньше одного периода сигнала, то полученные значения максимума и минимума могут резко отличаться от присущих сигналу на самом деле, поскольку при этом обрабатывается не весь сигнал (хотя бы его период), а только вырезка из него.

4.4.2. Блок вычисления среднего значения сигнала

Под *средним значением* сигнала подразумевается величина

$$mean = y_{\min} + \frac{y_{\max} - y_{\min}}{2}.$$

Это значение вычисляет блок Avg Value, модель которого представлена на рис. 4.25. Нетрудно убедиться, что блок реализует вычисление среднего значе-

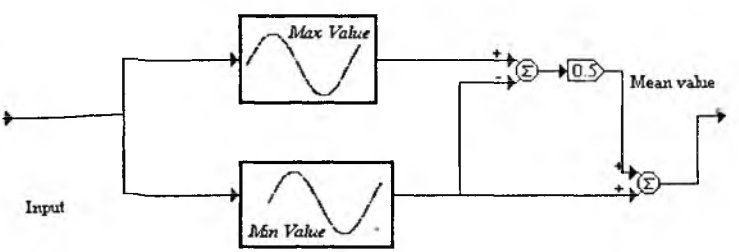


Рис. 4.25. Модель вычислителя среднего значения сигнала

ния по приведенному выше выражению и с применением ранее описанны блоков для вычисления минимального и максимального значений сигнала.

Пример вычисления среднего значения сигнала показан на рис. 4.2 Здесь в качестве сигнала применяется выпрямленная синусоида, для чего ис пользуется функция вычисления ее абсолютного значения.

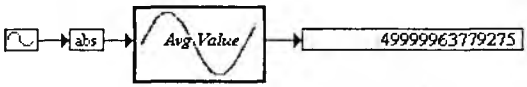


Рис. 4.26. Пример вычисления среднего значения сигнала

Подобный сигнал характерен для двухполупериодного выпрямителя пер менного тока, работающего на активную нагрузку. Если нагрузка емкостна то выходное напряжение может быть заметно выше среднего значения, по скольку конденсатор фильтра способен заряжаться до амплитудного значени выходного напряжения выпрямителя.

4.4.3. Блок вычисления периода

Период является важной характеристикой многих сигналов, в том чис синусоидальных. Если сигнал следует с частотой f, то период сигнала рав $T = 1/f$. Однако такое определение предполагает вычисление частоты, что н менее сложно, чем вычисление периода. Для неизвестного сигнала перио обычно вычисляется как интервал времени между повторяющимися послед вательностями сигнала.

Этот алгоритм реализует блок Period, модель которого представлена н рис. 4.27. Уровень сигнала задается по его среднему значению, для чего испо льзуется описанный выше блок измерения среднего значения. С помощь блока фиксации пересечения сигнала фиксируются моменты для перебро триггера, который создает сигнал, длительность которого равна периоду вхо ного сигнала. Заполнение сигнала триггера импульсами с эталонным перио дом позволяет получить численное значение периода входного сигнала.

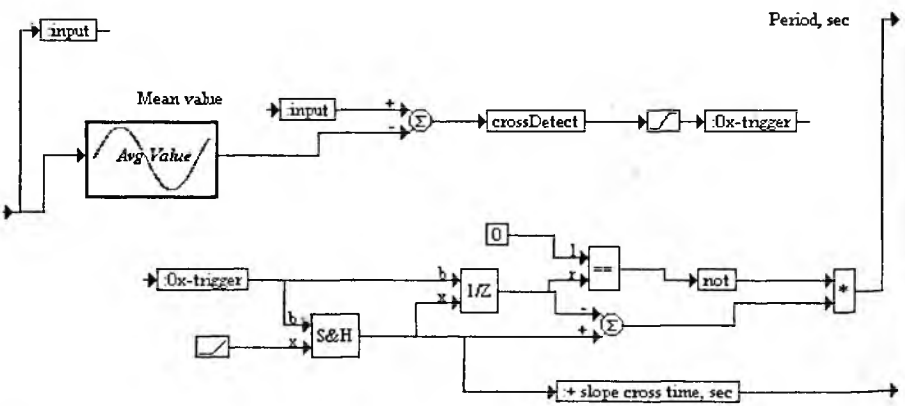


Рис. 4.27. Модель измерителя периода сигналов

Этот блок имеет один вход и два выхода. На одном выходе фиксируется момент первого переброса триггера (для синусоиды это 0), а на втором собственно длительность периода. Рисунок 4.28 показывает пример применения этого блока для измерения периода синусоидального сигнала.

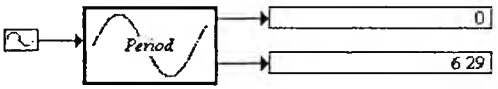


Рис. 4.28. Пример измерения периода синусоидального сигнала

Следует отметить, что реализованный в блоке Period алгоритм нельзя назвать безупречным. Он более или менее стабильно вычисляет период простых сигналов, таких как синусоида, меандр (симметричные прямоугольные волны), треугольные колебания и др. Однако если сигнал сильно зашумлен, то результаты вычислений периода могут оказаться недостоверными.

4.4.4. Совместное применение блоков контроля параметров сигналов

Специфика описанных выше блоков состоит в том, что они не являются полноценными библиотечными блоками и их нельзя вывести в любую модель, создаваемую пользователем. Для этих блоков нет ни кнопок в панелях инструментов, ни позиций в подменю Blocks. Блоки загружаются как файлы и как субблоки. Довольно легко можно ввести эти блоки в нужную модель пользователя, разместив загруженный из файла блок в буфер Windows и уже затем командой Paste переместив блок в модель пользователя.

Именно таким образом подготовлена демонстрационная модель, показанная на рис. 4.29. Здесь использовано сразу четыре измерительных блока, измеряющих параметры синусоидального сигнала.

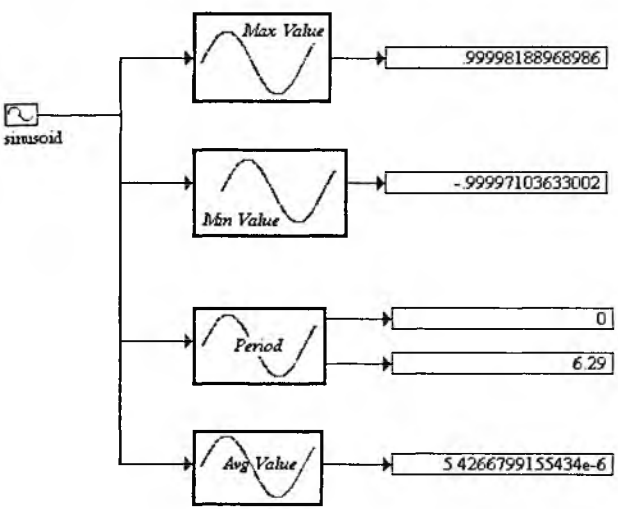


Рис. 4.29. Модельное измерение параметров синусоидального сигнала четырьмя измерительными блоками

4.4.5. Блок счетчика импульсов

При использовании импульсных сигналов часто возникает необходимост в подсчете числа импульсов. Это делают *счетчики импульсов*. Блок Pulse Coun ter, модель которого приведена на рис. 4.30, создает выходной сигнал, кото рый увеличивается на заданную величину (1 по умолчанию) при каждом по ступлении на вход блока импульса.

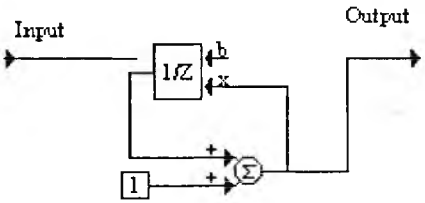


Рис. 4.30. Блок счетчика импульсов

Как нетрудно заметить из рис. 4.31, блок при подаче на его вход непр рывной последовательности импульсов создает ступенчато нарастающий си нал, причем амплитуда ступеньки определяется числом поступивших на вх импульсов к заданному моменту времени. Такая работа блока позволяет испо льзовать блок в качестве генератора ступенчатых сигналов, релаксационны генераторов, делителей частоты и ряда других устройств.

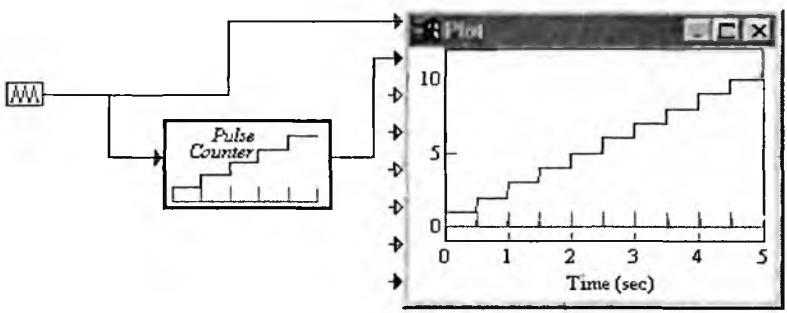


Рис. 4.31. Применение блока счетчика импульсов

4.4.6. Блок измерения среднеквадратического значения

Для контроля *среднеквадратического значения* сигнала служит блок rm (или RMS — Root Mean Square). Модель этого блока представлена н рис. 4.32. Блок имеет два входа. На первый вход подается постоянный п уровню сигнал или перепад, задающий начало вычислений среднеквадратич ского значения сигнала, а на второй вход уже сам сигнал.

Рисунок 4.33 показывает применение блока rms для вычисления текущег среднеквадратического значения синусоидального сигнала. Подача на вер ний вход константы +1 означает, что вычисления начинаются с начала мод лирования.

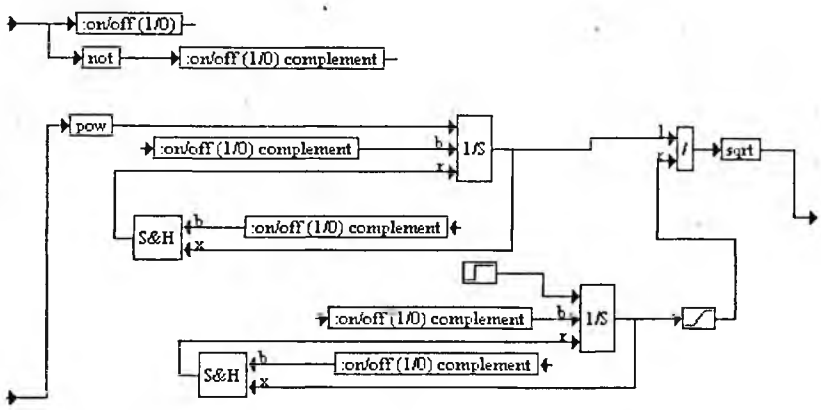


Рис. 4.32. Модель блока для вычисления среднеквадратического значения сигнала

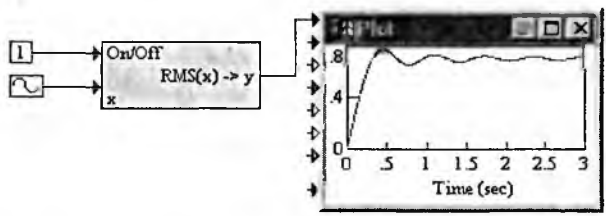


Рис. 4.33. Применение блока rms для вычисления текущего среднеквадратического значения синусоидального сигнала

4.4.7. Блок измерения отношения амплитуд и разности фаз сигналов

Как было отмечено в главе 1, при моделировании линейных систем широко используются амплитудно-частотные (АЧХ) и фазо-частотные (ФЧХ) характеристики таких систем. Для измерения АЧХ нужно вычислять отношение сигналов на выходе и входе системы (*магнитуду* — magnitude), а для измерения ФЧХ вычислять разность фаз сигналов. Обычно это делается в определенном диапазоне частот синусоидального тестового сигнала.

Блок mag_phas служит для измерения отношения амплитуд и разности фаз двух сигналов, подаваемых на два входа этого блока. Модель блока представлена на рис. 4.34. На верхний вход R подается исходный сигнал, а на нижний S — второй сигнал. Вычисления производятся с заданной погрешностью Epsilon (по умолчанию 0,0001).

Блок имеет два выхода — на одном создается сигнал отношения, на другом — разность фаз. Пример применения блока для вычисления этих параметров двух синусоидальных сигналов с одинаковой частотой представлен на рис. 4.35. Для создания фазового сдвига второй сигнал задан с временной задержкой. Фазовый сдвиг вычисляется с приведением в диапазон углов от −180° до +180°.

Этот блок широко используется в практике экспресс-анализа линейных цепей, что описано в главе 5.

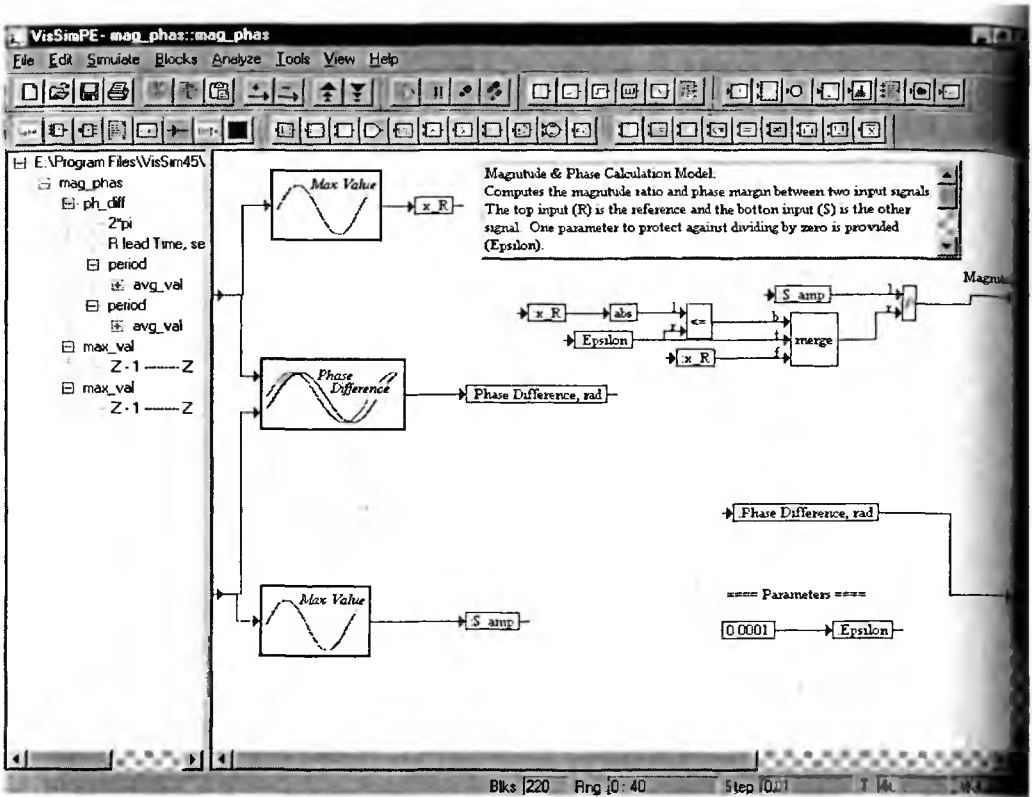


Рис. 4.34. Модель блока mag_phas для измерения отношения амплитуд и сдвига фаз двух сигналов

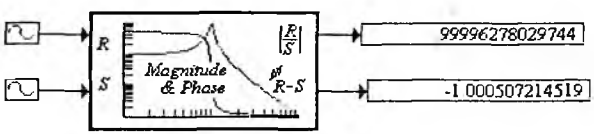


Рис. 4.35. Пример применения блока mag_phas

4.5. Блоки раздела Siggen

В папке (разделе) Siggen (сигнал-генераторы) находится несколько фа лов с моделями блоков контроля времени и генерации сигналов. Ниже пред ставлено их описание.

4.5.1. Блок вычисления шага моделирования

Блок вычисления *шага моделирования* имеет модель, представленную н рис. 4.36. Шаг моделирования dt вычисляется с разрядностью, заданной зна чением переменной First Past Value, которое принято равным 10^{-6}. Приме вычисления шага моделирования дан на рис. 4.37.

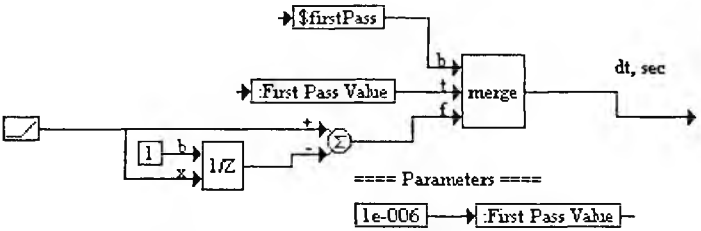


Рис. 4.36. Модель блока вычисления шага моделирования

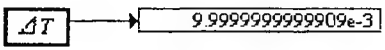


Рис. 4.37. Пример вычисления шага моделирования

4.5.2. Блок выдачи времени

Для *выдачи времени* (в формате минут, часов и дней) служит блок Calendar. Основная модель этого блока представлена на рис. 4.38. Текстовое окно в блоке содержит хотя и англоязычное, но вполне понятное описание формата выходных данных этого блока. Основная модель содержит три подмодели Calulate Minute (Вычисление минуты), Calulate Hour (Вычисление часа) и Calulate Day (Вычисление дня). Соответственно, блок имеет три выхода для вывода вычисленного текущего времени.

Подмодель вычисления текущей минуты показана на рис. 4.39. Вычисление минут основано на применении в качестве счетчика секундных импульсов квантователя. Предусмотрена инициализация секунд, указывающая, с ка-

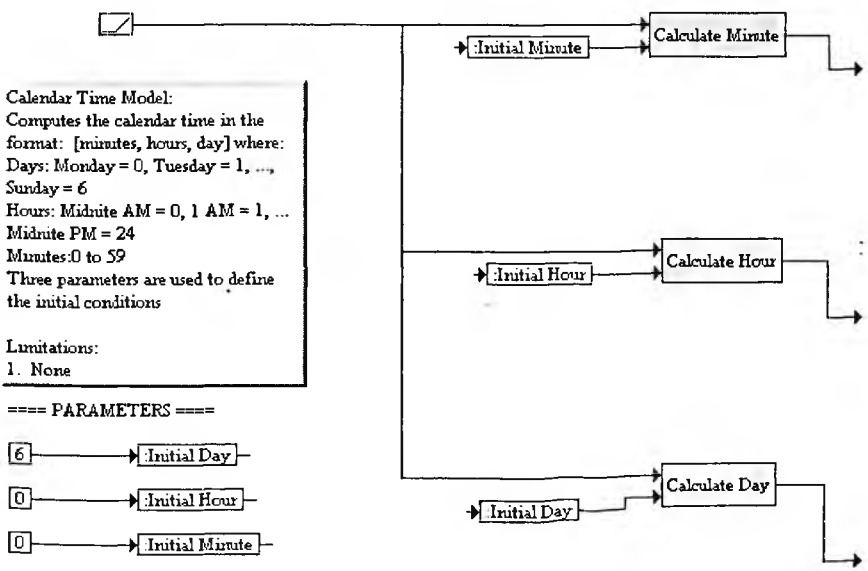


Рис. 4.38. Основная модель календаря Calendar

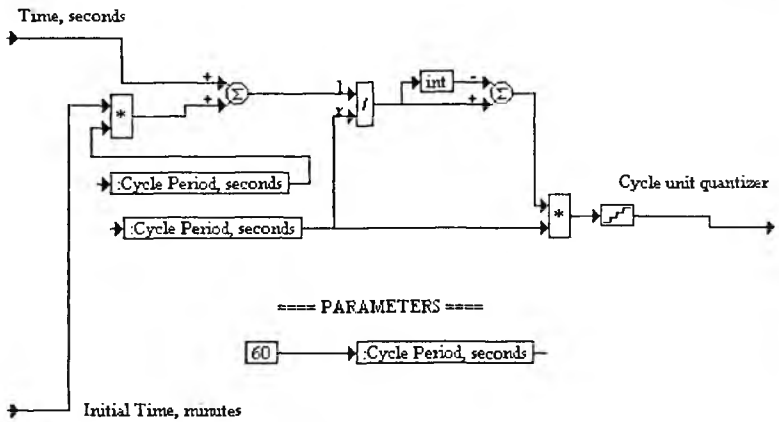


Рис. 4.39. Подмодель вычисления минуты

кого значения времени начинается отсчет секунд. По умолчанию этот пар метр задан нулевым (см. список параметров инициализации в основно модели, рис. 4.38).

Подмодель вычисления текущего часа представлена на рис. 4.40. Здес предусмотрена инициализация минут, указывающая, с какого значения нач нается отсчет часа. По умолчанию этот параметр задан нулевым — см. спис параметров инициализации в основной модели, рис. 4.38).

Подмодель вычисления текущего дня показана на рис. 4.41. Здесь пред смотрена инициализация дня, указывающая, с какого значения начинает отсчет дня. По умолчанию этот параметр задан равным 6 (см. список пар метров инициализации в основной модели, рис. 4.38).

Пример применения блока календаря представлен на рис. 4.42. Этот бл обычно имеет смысл применять при моделировании в реальном масшта времени. В противном случае он имеет чисто информационное значение.

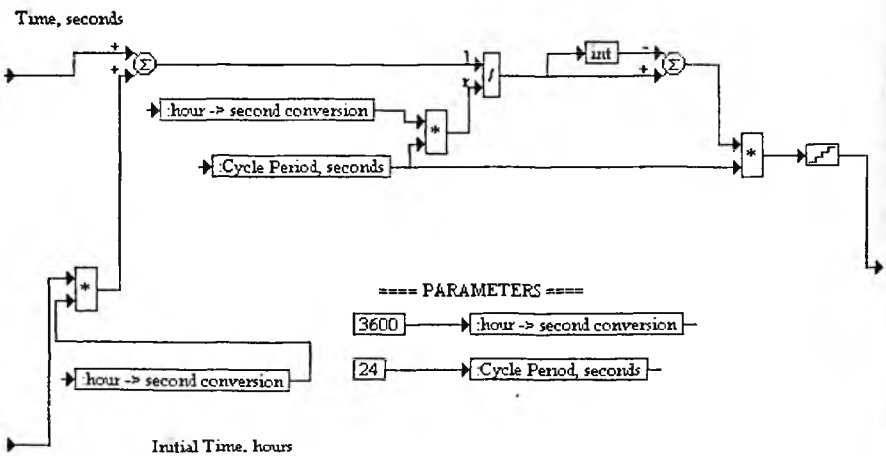


Рис. 4.40. Подмодель вычисления часа

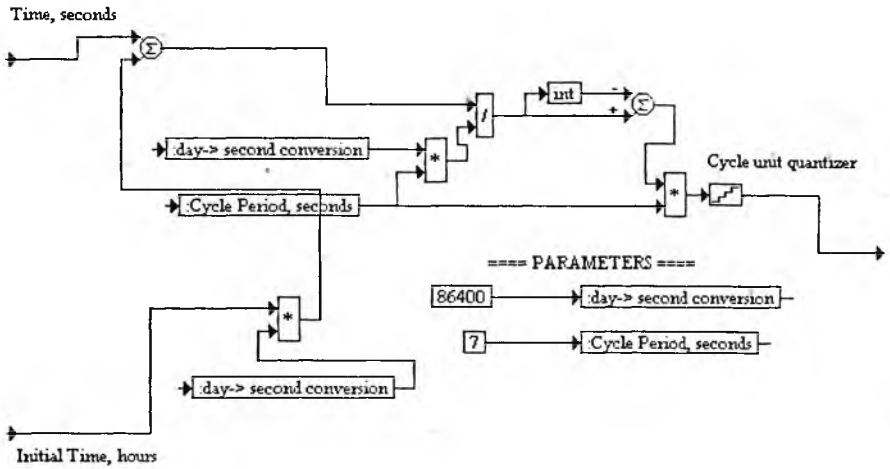


Рис. 4.41. Подмодель вычисления дня

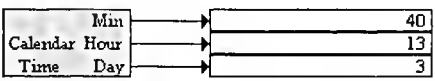


Рис. 4.42. Пример работы блока календаря

4.5.3. Блок генерации пилообразного сигнала

Пилообразные сигналы находят весьма широкое распространение в преобразователях «напряжение—время», генераторах развертки осциллографов, телевизионных дисплеях и других устройствах. Хорошо известна возможность формирования пилообразного сигнала с помощью интегратора. Действительно, если входной сигнал интегратора постоянен x = X0, то

$$y(t) = \int X_0 dt = y(0) + X_0 \cdot t.$$

Блок Sawtooth Wave использует интегратор со сбросом для генерации пилообразного сигнал начиная с нулевого значения $y(0) = 0$. Предусмотрена установка периода колебаний Period (в секундах) и амплитуды пилообразного сигнала MaxVal. Реализация модели генератора дана на рис. 4.43.

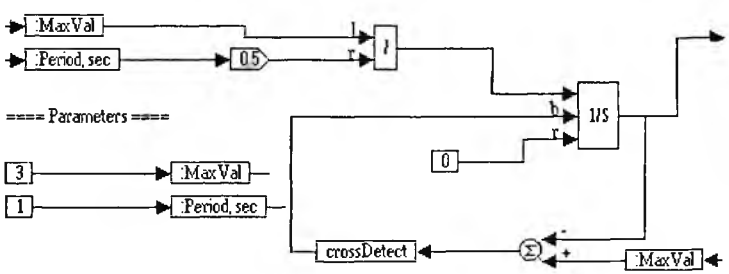


Рис. 4.43. Модель генератора пилообразных сигналов

Пример применения блока Sawtooth Wave представлен на рис. 4.44. На языке импульсной техники данный генератор обеспечивает функции автоколебательного генератора пилообразных импульсов. Ждущий режим его работы, как и вход для импульсов запуска, не предусмотрен.

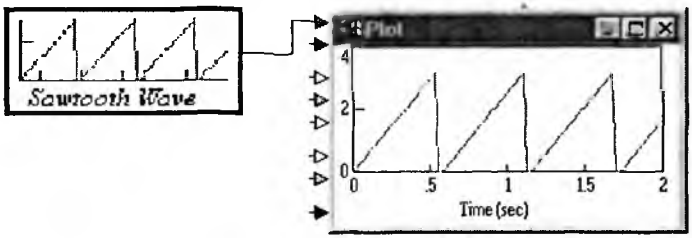


Рис. 4.44. Демонстрация работы генератора пилообразных сигналов

4.5.4. Блок генерации прямоугольных несимметричных сигналов

Генераторы прямоугольных сигналов являются наиболее распространенными тестовыми устройствами. Они используются для снятия переходных характеристики и запуска различных устройств. Блок Square Wave служит для генерации прямоугольных несимметричных импульсов. Модель блока представлена на рис. 4.45.

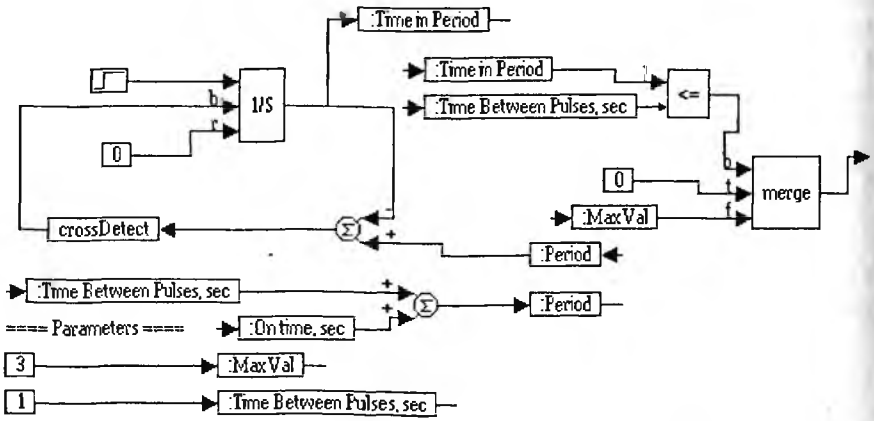


Рис. 4.45. Модель генератора прямоугольных импульсов

В модели генератора можно задавать два параметра — максимальное значение MaxVal, задающее амплитуду импульсов, и время между импульсами Time Between Pulses (в секундах), задающее степень асимметрии импульсов. Для наблюдения за формой импульсов на выходе блока, как обычно, можно использовать осциллограф (рис. 4.46).

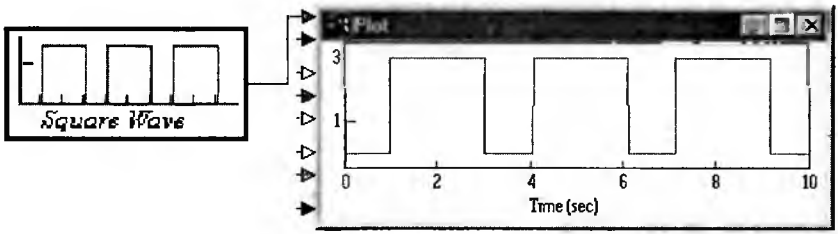


Рис. 4.46. Демонстрация работы прямоугольных импульсов

4.5.5. Блок генерации треугольных колебаний

Для генерации треугольных колебаний с равными времена нарастания и спада используется блок Triangle Wave, модель которого представлена на рис. 4.47. Для получения такого сигнала используется интегратор, на вход которого подан разнополярный сигнал. Если он положительный, то выходной сигнал растет по линейному закону, а если отрицательный, то он падает по линейному закону. Чтобы получить привязку выходного сигнала к нулю, на выходе интегратора установлен блок получения абсолютного значения сигнала.

Модель генератора треугольных колебаний имеет два параметра настройки — максимальный уровень сигнала MaxVal и период колебаний Period (в секундах). Демонстрация работы генератора представлена на рис. 4.48.

Реализация этого блока не совсем доработана. Как видно из рис. 4.48, имеет место небольшое подрезание вершины треугольных импульсов.

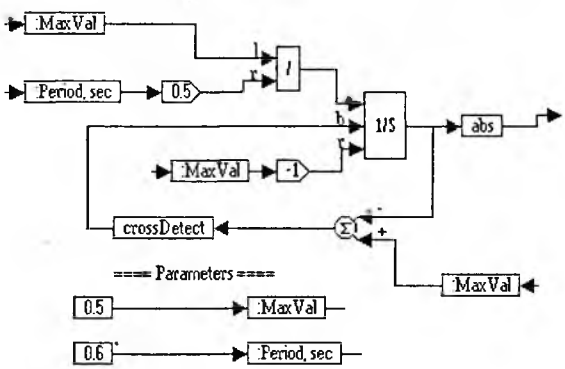


Рис. 4.47. Модель генератора треугольных колебаний

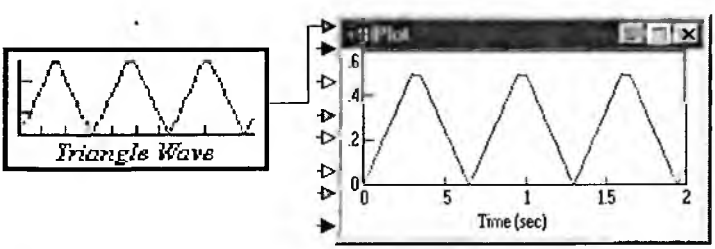


Рис. 4.48. Демонстрация работы генератора треугольных колебаний

4.5.6. Блок генерации трехфазного синусоидального сигнала

В промышленных сетях переменного тока широко используются трехфазные синусоидальные сигналы. Правильнее их называть напряжениями или токами, поскольку они выполняют функцию передачи энергии, а не информации, что характерно для сигналов. Тем не менее ради однообразия сохраним этот термин и за выходным параметром блока 3 Phase Generator. Модель этого блока представлена на рис. 4.49.

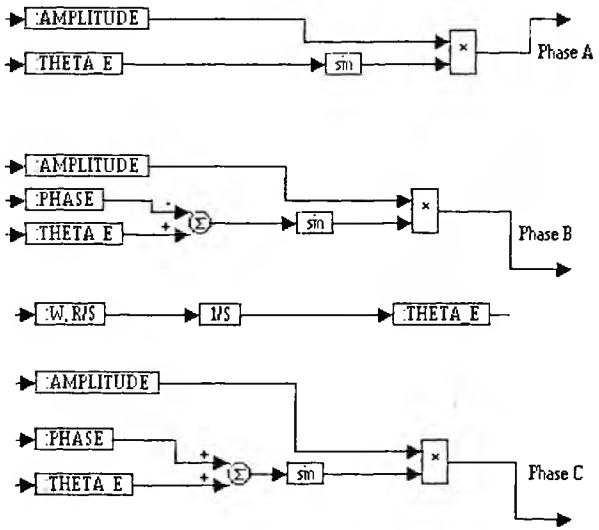


Рис. 4.49. Модель генератора трехфазных синусоидальных сигналов

Демонстрация работы генератора трехфазных сигналов дана на рис. 4.50. Нетрудно заметить, что сдвиг по фазе одного из сигналов составляет треть периода, а другого — две трети периода по сравнению с опорным сигналом.

Применение трехфазных сигналов в электроэнергетике связано с повышенным к.п.д. трехфазных устройств и пониженными требованиями к фильтрации сглаженного напряжения на выходе трехфазного выпрямителя.

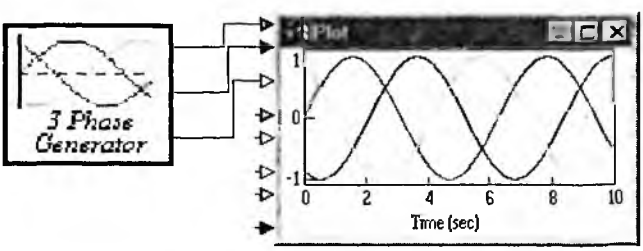


Рис. 4.50. Демонстрация работы генератора трехфазных синусоидальных сигналов

4.6. Специальные методы генерации сигналов

4.6.1. Генерация сигналов на основе комбинаций элементарных функций

Как это ни странно, но в составе блоков генерации сигналов в VisSim нет целого ряда широко применяемых сигналов, например симметричных двухполярных прямоугольных импульсов (меандра), симметричных двухполярных треугольных импульсов и др. Кроме того, генерация ряда сигналов в блоках, описанных в предшествующем разделе, отличается сложностью, что может привести к нестабильному процессу моделирования.

Между тем есть простой способ генерации большинства тестовых сигналов на основе *комбинаций элементарных функций* и применения некоторых стандартных функций системы VisSim. Наиболее характерные из таких возможностей представлены ниже:

- sin(t) — периодическая синусоидальная функция;
- $\sin(t)^n$ — при нечетном n похожая на синус функция с полочками;
- sign(sin*t)) — прямоугольный симметричный импульс (меандр);
- |sin(t)| — выпрямленная синусоида (сигнал двухполупериодного выпрямителя);
- asin(sin(t)) — треугольный симметричный сигнал;
- tan(sin(t)) — треугольный симметричный сигнал со скругленными верхушками;

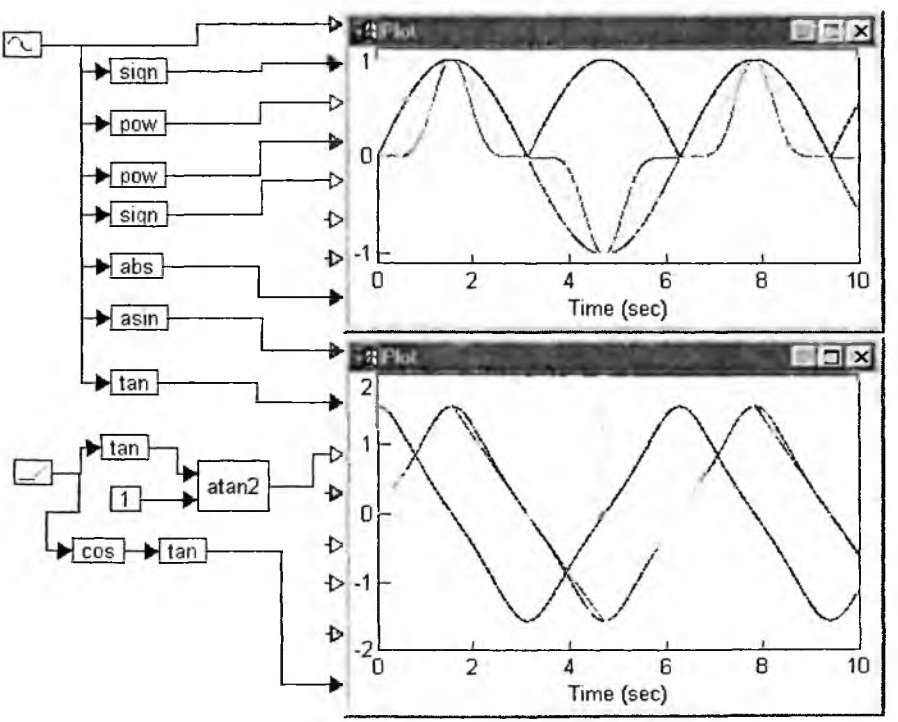


Рис. 4.51. Создание сигналов различного вида на основе комбинаций элементарных функций

• atan(tan(t)) — пилообразный сигнал с удвоенной частотой повторения;
• tan(cos(x)) — почти треугольный сигнал со сглаженными верхушками.

Рисунок 4.51 показывает реализацию этих возможностей с помощью бл ков системы VisSim. В большинстве случаев эта реализация намного про той, что была описана в предшествующем разделе.

Большие возможности в создании сигналов с помощью как представле ных комбинаций элементарных функций, так и других произвольных завис мостей дает блок expression, позволяющий задать любую функциональную з висимость для сигналов.

4.6.2. Генерация коротких сдвинутых попарно импульсов

Иногда возникает необходимость в генерации коротких разнополярны попарно сдвинутых импульсов, симметрично расположенных относитель экстремумов базового синусоидального сигнала. Модель, представленная н рис. 4.52, иллюстрирует реализацию этой возможности. Входящий в моде субблок представлен на рис. 4.53.

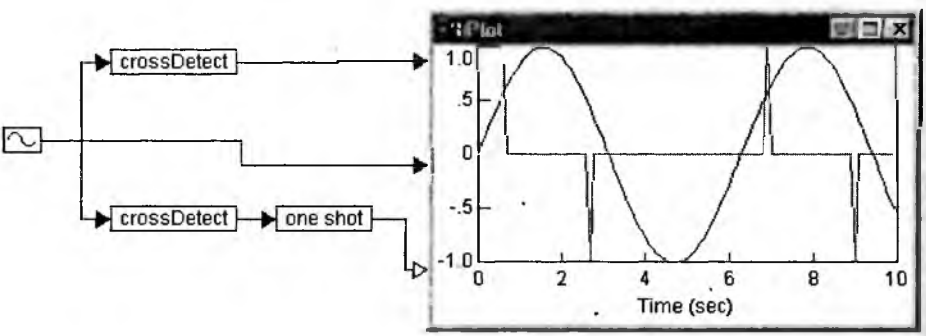


Рис. 4.52. Генерация коротких разнополярных импульсов, симметрично расположенных относительно экстремумов синусоиды

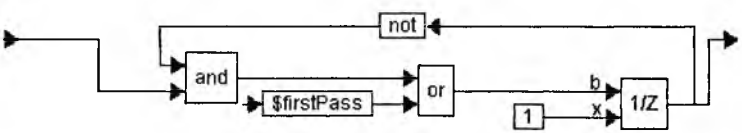


Рис. 4.53. Субблок one shot, входящий в модель рис. 4.52

4.6.3. Амплитудная модуляция

Синусоидальные сигналы с амплитудной модуляцией (АМ) широко пр меняются в технике радиосвязи. Достаточно отметить, что радиовещание н длинных, средних и коротких волнах ведется с применением АМ. Амплиту но-модулированный нормированный сигнал описывается выражением:

$$e(t) = (1 + m \cdot (\sin(2\pi Ft)) \cdot \sin(2\pi ft),$$

где m — коэффициент модуляции; F — частота модуляции и f — частота несущей (обычно $F \ll f$).

Из этого выражения нетрудно получить выражение для спектра АМ-сигнала:

$$e(t) = 0.5 \cdot m \cdot \sin(2\pi(f - F)t) + \sin(2\pi ft) + 0.5 \cdot m \cdot \sin(2\pi(f + F)t).$$

Отсюда видно, что АМ-сигнал имеет спектр из трех составляющих — центральной частоты и двух боковых частот с уровнями амплитуды $m/2$. Боковые частоты отстоят от несущей на расстояни, равном F.

Из сказанного вытекает два основных способа реализации моделей амплитудной модуляции — путем прямой модуляции несущего колебания низкочастотным (модулирующим) сигналом и путем синтеза АМ-колебаний по его спектру. Оба эти способа иллюстрируют простые модели, показанные на рис. 4.54.

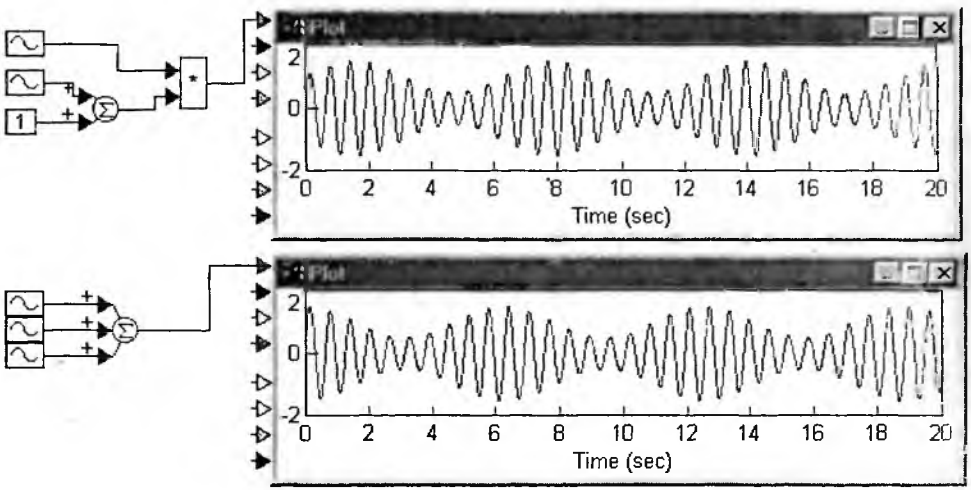


Рис. 4.54. Модели получения амплитудно-модулированного колебания с 50%-й модуляцией

Оба варианта дают идентичные результаты. Модель с прямой реализаций модуляции физически более наглядна, тогда как модель на основе спектрального синтеза чуть проще.

4.6.4. Частотная модуляция

Широко применяется также частотная модуляция синусоидальных сигналов. Такие сигналы можно ограничивать по амплитуде, что является простым способом создания широтно-импульсной модуляции и позволяет избавляться от помех, связанных с изменением амплитуды сигналов. Частотная модуляция используется в УКВ радиовещании. Она соответствует сигналу следующего вида:

$$e(t) = \sin(2\pi f(1 + k \cdot \sin(2\pi Ft)) \cdot t).$$

Параметр $k < 1$ определяет степень отклонения частоты от среднего значения f. Это отклонение происходит с частотой модуляции $F \ll f$. Одна из возможных моделей частотной модуляции представлена на рис. 4.55.

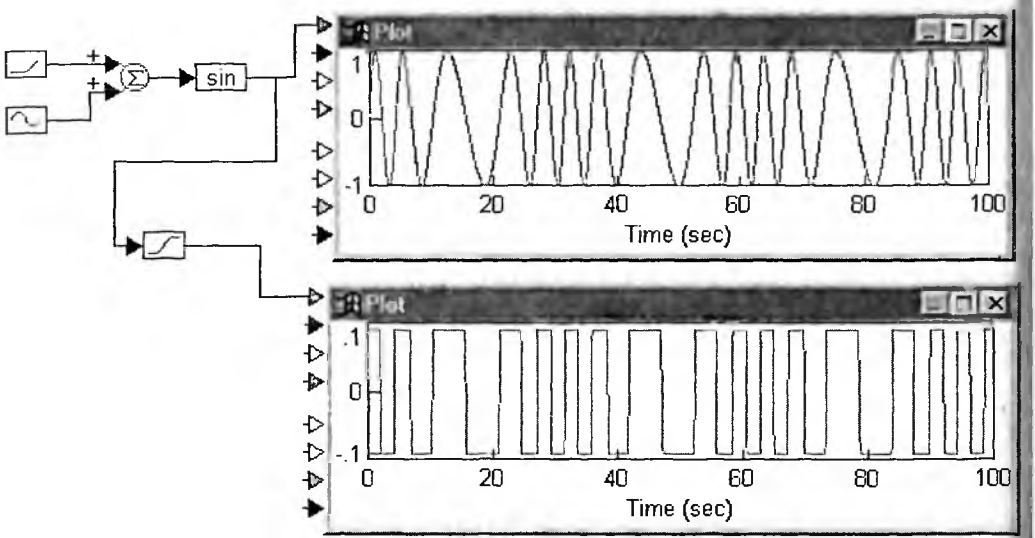


Рис. 4.55. Модель созданиям частотно-модулированного сигнала

На рис. 4.55 показан также пример применения ограничителя уровня частотно-модулированного сигнала. При глубоком ограничении (на уровне ±0,1 в приведенном примере) формируются прямоугольные импульсы с изменяющейся частотой.

4.6.5. Широтно-импульсная модуляция

Широтно-импульсной модуляцией импульсных (для определенности прямоугольных) сигналов называется изменение во времени ширины импульсов, или, что то же самое, скважности импульсов. Импульсы с такой модуляцией прекрасно подходят для управления силовыми ключами в преобразовательных устройствах промышленной и силовой электроники. Блок PWM реализует этот вид модуляции. Его модель представлена на рис. 4.56.

Модель модулятора имитирует работу типичного релаксационного генератора на основе триггера и интегратора со сбросом. На рис. 4.57 представлен субблок этой модели PULSE TRAIN.

Демонстрация работы широтно-импульсного модулятора представлена на рис. 4.58. Здесь модулирующим сигналом является синусоида. На пике синусоиды импульсы имеют максимальную ширину (скважность около двух), а на впадине — минимальную ширину. Если модулятор управляет силовым ключом, то это означает отдачу максимальной мощности в пике синусоиды и минимальной в ее впадине.

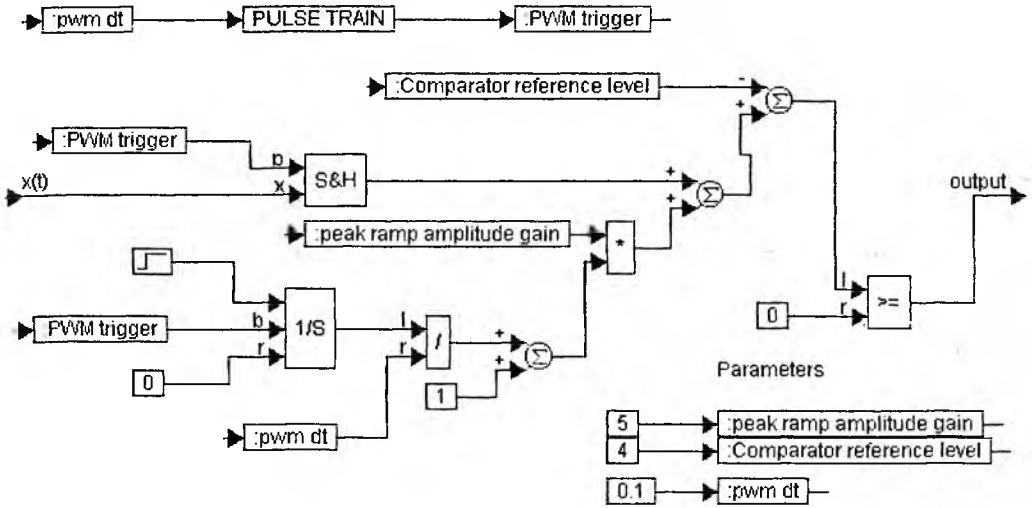


Рис. 4.56. Модель широтно-импульсного модулятора

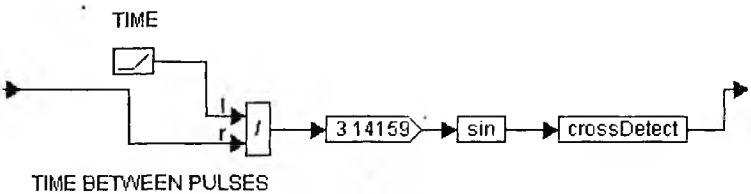


Рис. 4.57. Субблок PULSE TRAIN

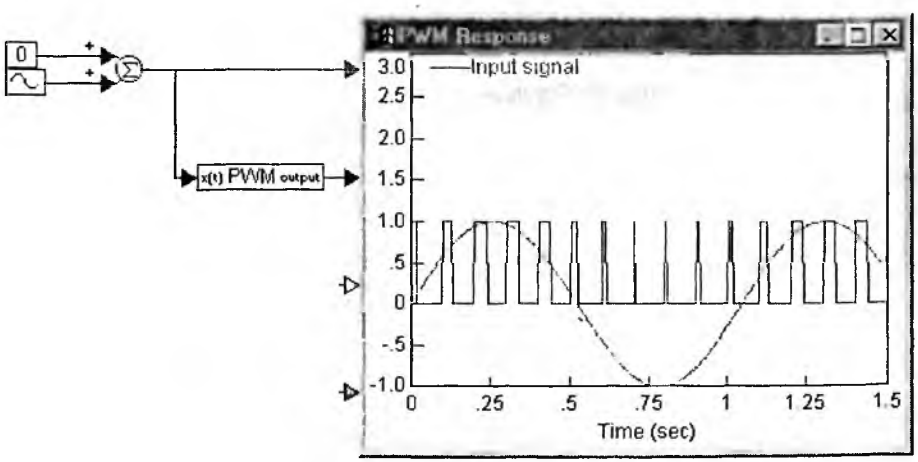


Рис. 4.58. Демонстрация работы широтно-импульсного модулятора

4.6.6. Генерация зашумленных сигналов

Как уже отмечалось, библиотека VisSim содержит три генератора шумовых сигналов: генератор шума $e_ш$ с равномерным в диапазоне [0,1] распределением, генератор Гаусса шума и генератор псевдослучайного бинарного сигнала. Прибавление к шуму с равномерным распределением константы –1 обеспечивает устранение постоянной составляющей этого шума и позволяет создать так называемый «белый» шум.

Наряду с самостоятельным применением источников шума часто используется зашумленный сигнал той или иной формы, например синусоидальной. Различают два основных вида зашумленных сигналов: аддитивные и мультипликативные. В аддитивных зашумленных сигналах компоненты сигнала $e_с$ и шума $e_ш$ складываются, т. е. сигнал представляется в виде:

$$e(t) = e_с(t) + e_ш(t).$$

Для изменения уровня соотношения сигнал/шум можно изменять уровни сигнала и шума в окнах свойств либо предусмотреть введение блоков gain в каналы сигнала и шума. Рисунок 4.59 иллюстрирует различные способы создания зашумленных сигналов. Пример сверху показывает создание аддитивного зашумленного сигнала в виде синусоидального сигнала с примесью шумовой компоненты.

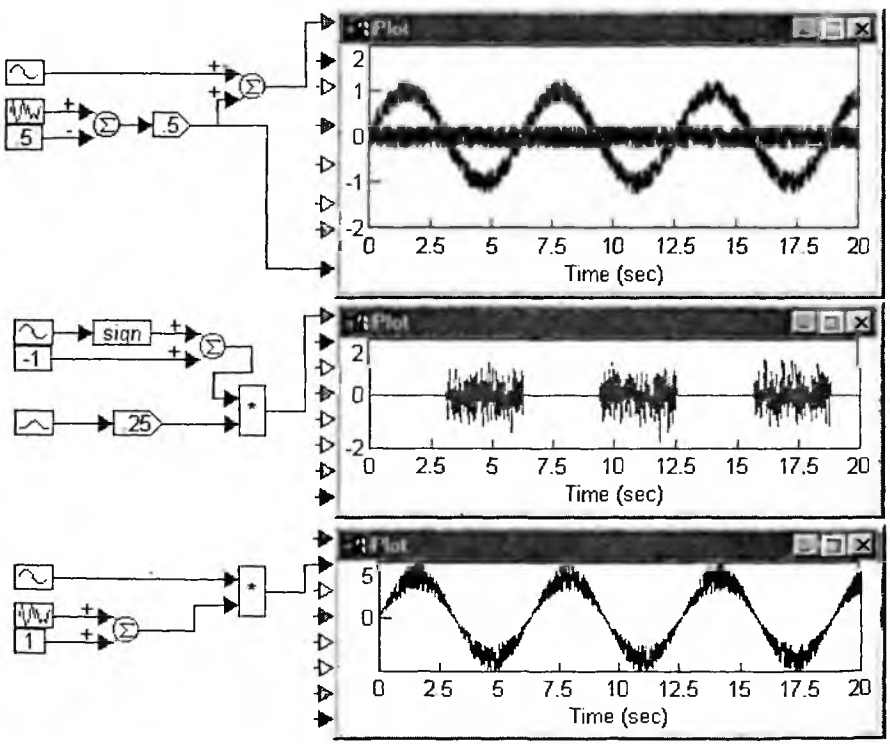


Рис. 4.59. Некоторые примеры создания зашумленных сигналов

Мультипликативный шум соответствует перемножению сигнальной и шумовой компонент:

$$e(t) = (a + b \cdot e_{ш}(t)) \cdot e_{с}(t).$$

Константа a обеспечивает изменение уровня постоянной составляющей шума, а константа b позволяет менять уровень шума. В данном случае мы фактически имеем модуляцию сигнала шумом. В средней части рис. 4.59 представлен пример создания пачек шумовых сигналов. Используется перемножение сдвинутого на величину +1 меандра с Гаусовым шумом. Поскольку сдвинутый меандр попеременно имеет значения 0 и +2, то формируются пачки шума (в течение одного полупериода сигнал равен 0, а в течение второго полупериода он является «чистым» шумом).

В нижней части рис. 4.59 показан пример создания мультипликативного сигнала, компонентами которого являются синусоида и шум с нормальным распределением, имеющим постоянную составляющую $a = 1$. Сигнал явно представляет собой промодулированную шумом синусоиду.

4.7. Математическая обработка сигналов

4.7.1. Фурье-синтез прямоугольного импульса с наклонной вершиной

Одним из достаточно универсальных способов создания сигналов является их Фурье-синтез из достаточно большого числа гармоник — синусоидальных сигналов с частотами, кратными частоте повторения импульсов. При определенном подборе амплитуд и фаз гармоник можно синтезировать сигналы той или иной формы. В папке Sig_proc системы VisSim имеется пример F_series на синтез прямоугольного импульса с наклонной вершиной. Пример, прямо скажем, слишком переусложненный и не вполне удачный. Поэтому мы рассмотрим его кратко.

Основная модель примера F_series представлена на рис. 4.60. Здесь же показан результат синтеза импульса по наборам (термам) из 5, 10 и 15 гармоник. Как видно из этого рисунка, даже 15 гармоник явно недостаточно, чтобы синтезированное колебание достаточно точно представляло импульс, который создан как результат наложения на прямоугольный импульс пилообразного импульса. В частности, хорошо заметен эффект Гиббса в виде отчетливо видных колебаний, особенно больших в точках разрывов и перегибов синтезируемой функции.

Как видно из рис. 4.60, в данном примере синтез импульса по гармоникам реализован с помощью ряда субблоков, внутри которых имеются и субблоки более низкого уровня. Читатель может познакомиться с их построением, но, поскольку данный пример реализует гармонический синтез далеко не самым лучшим и прямым способом, мы не будем описывать все эти блоки — заинтересованный читатель может ознакомиться с ними самостоятельно.

Гораздо проще синтезировать подобное нужное колебание как совокупность импульсных временных зависимостей. Это и делает блок f(t) waveform, диаграмма которого приведена на рис. 4.61.

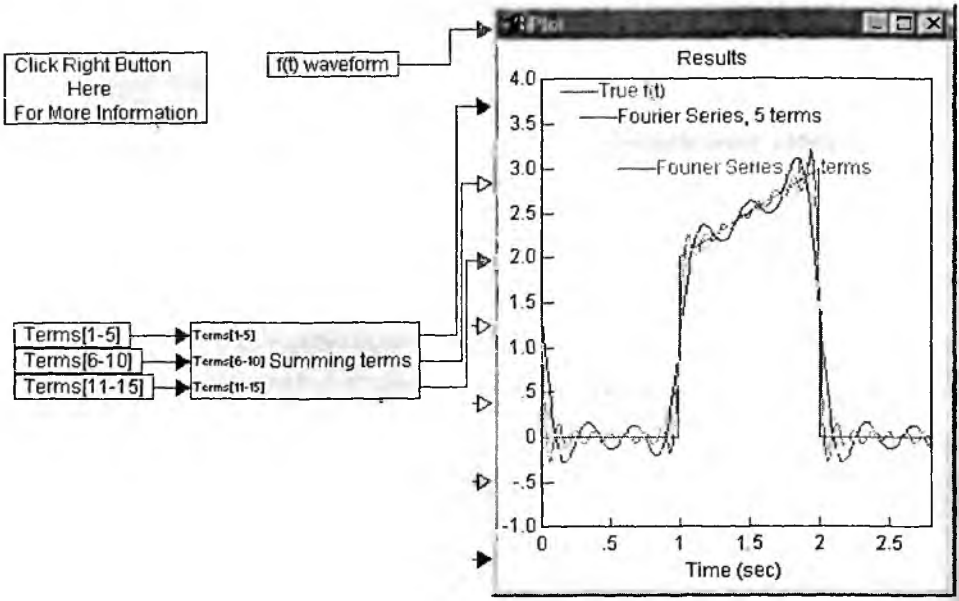


Рис. 4.60. Пример Фурье-синтеза прямоугольного импульса с наклонной вершиной

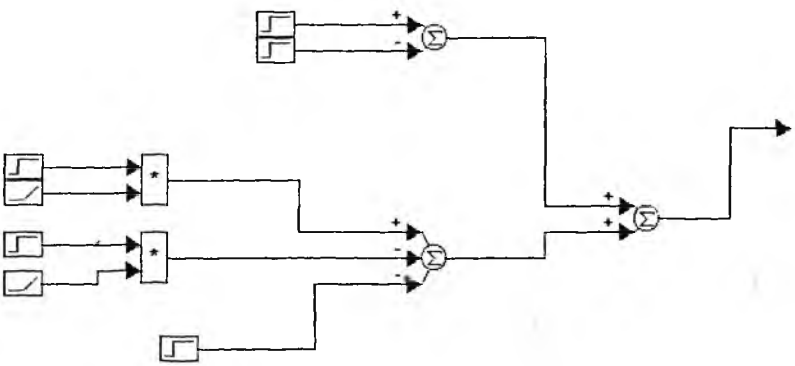


Рис. 4.61. Пример прямого синтеза прямоугольного импульса с наклонной вершиной

В дальнейшем мы опишем более простые и удобные способы гармонического синтеза импульсных сигналов.

4.7.2. Реализация свертки

Для получения реакции линейной системы на заданное входное воздействие может использоваться интеграл свертки (см. раздел 1.6.5). В файле Comvolution системы VisSim представлен пример применения интеграла свертки в виде:

$$y(t) = h(t) \bullet x(\tau) = \int_0^t h(t-\tau) \cdot x(\tau) d\tau.$$

Здесь $h(t)$ — импульсная характеристика (не спутайте с переходной), а $x(t)$ — входное воздействие. Могут вызвать недоумение пределы интегрирования, поскольку в классической записи выражения для свертки они равны $-\infty$ и $+\infty$. Однако создатели примера учли условия его реализации — равенство 0 $h(t)$ при $t < 0$ и моделирование в пределах от 0 до времени t.

В упомянутом примере выполняется сравнение двух методов вычисления реакции системы с заданной передаточной функцией — аналитического, основанного на преобразовании Лапласа, и численного. Основная модель примера представлена на рис. 4.62. Она содержит прямоугольники с субблоками и окно осциллографа, в котором и дано соответствующее сравнение.

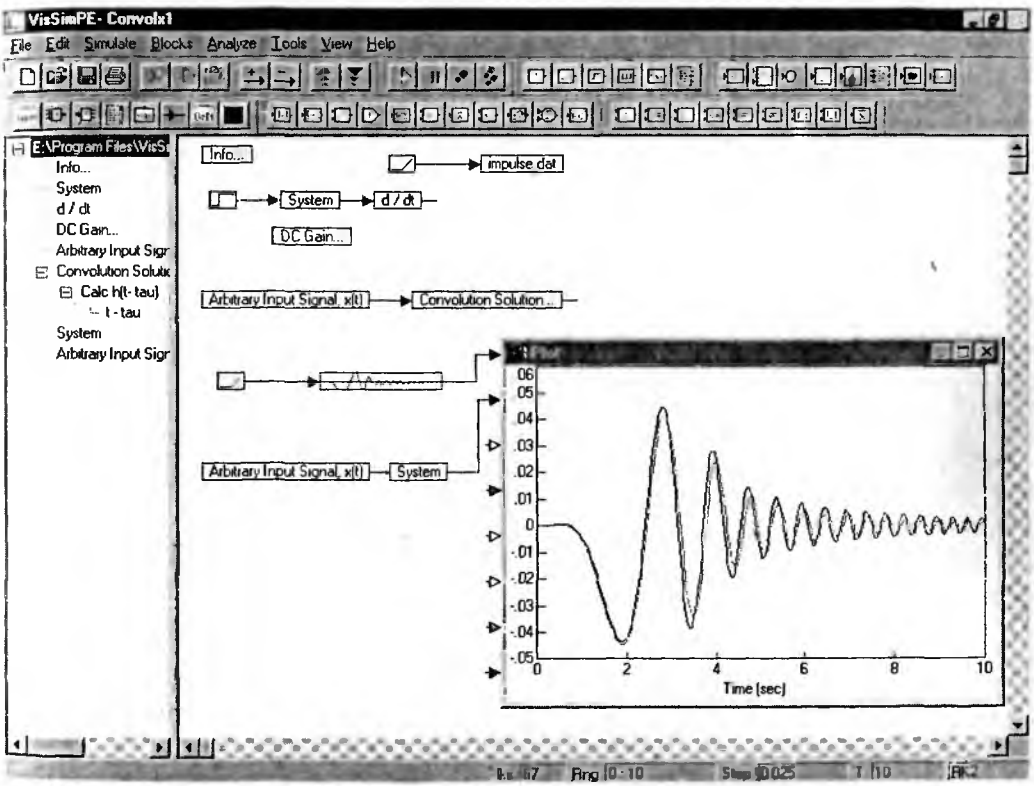


Рис. 4.62. Основная модель оценки реакции системы по интегралу свертки

Субблок задания неминимально-фазовой передаточной функции системы System представлен на рис. 4.63. Субблок задания производной d/dt не рассматривается, поскольку он уже описывался. Вспомогательный блок DC Gain демонстрирует равенство нулю постоянной составляющей сигнала, записанного в файле IMPULSE.DAT.

Субблок аналитического задания сигнала Arbitrary Input Signal x(t) представлен на рис. 4.64. Формируемый субблоком сигнал является типичным нестационарным сигналом, у которого во времени меняется как амплитуда, так и частота. Изменение частоты задается по квадратичному закону, а амплитуды — соответствующей передаточной функцией. Этот текстовый сигнал полу-

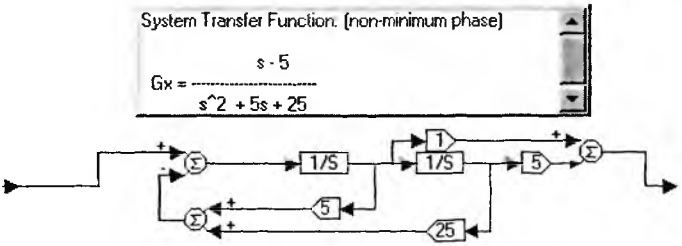


Рис. 4.63. Субблок задания передаточной функции системы

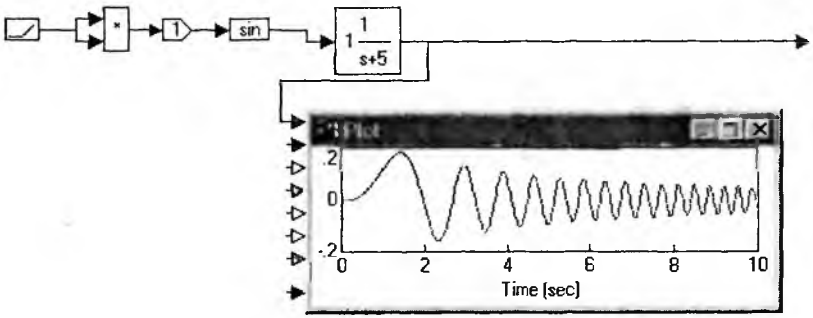


Рис. 4.64. Субблок создания сигнала «визг»

чил образное название «визг», поскольку при его прослушивании через акустические устройства действительно напоминает визг.

Субблок вычисления свертки с его субблоками более низкого уровня представлен на рис. 4.65—4.67. В соответствии с определением свертки реализация этих субблоков очевидна. В субблоке t-tau есть возможность подключения блоков останова Stop для досрочной остановки процесса моделирования (для чего это надо, в примере не комментируется).

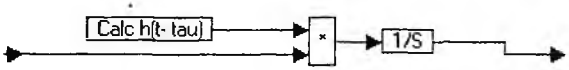


Рис. 4.65. Субблок вычисления свертки

Рис. 4.66. Субблок Calc h(t-tau)

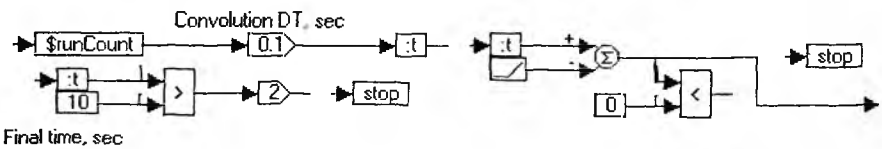


Рис. 4.67. Субблок t-tau

Как видно из рис. 4.62, два сопоставляемых метода использования операции свертки (аналитический и численный) дают практически идентичные результаты при расчете реакции системы на достаточно сложный нестационарный сигнал типа «визг».

4.7.3. Построение графика функции двух переменных — «шляпы» Sombrero

Сигналы изображений обычно представлены в виде функции двух переменных, например $z = f(x, y)$. Существуют разные формы представления таких функций. Например, учли z — это уровень серого цвета, то мы имеем представление монохромного графика, а если z — это высота поверхности, то можно говорить о представлении 3D-графика поверхности.

VisSim не имеет серьезных средств для построения 3D-графиков, подобных тем, что строят СКМ Mathcad и др. Тем не менее построить каркасный график поверхности можно. Это и иллюстрирует модель, показанная на рис. 4.68.

Эта модель имеет два субблока. Первый Y Sweep Calc обеспечивает формирование значений координаты y, которая должна смещаться после завершения формирования ряда значений координаты x. Это необходимо для фор-

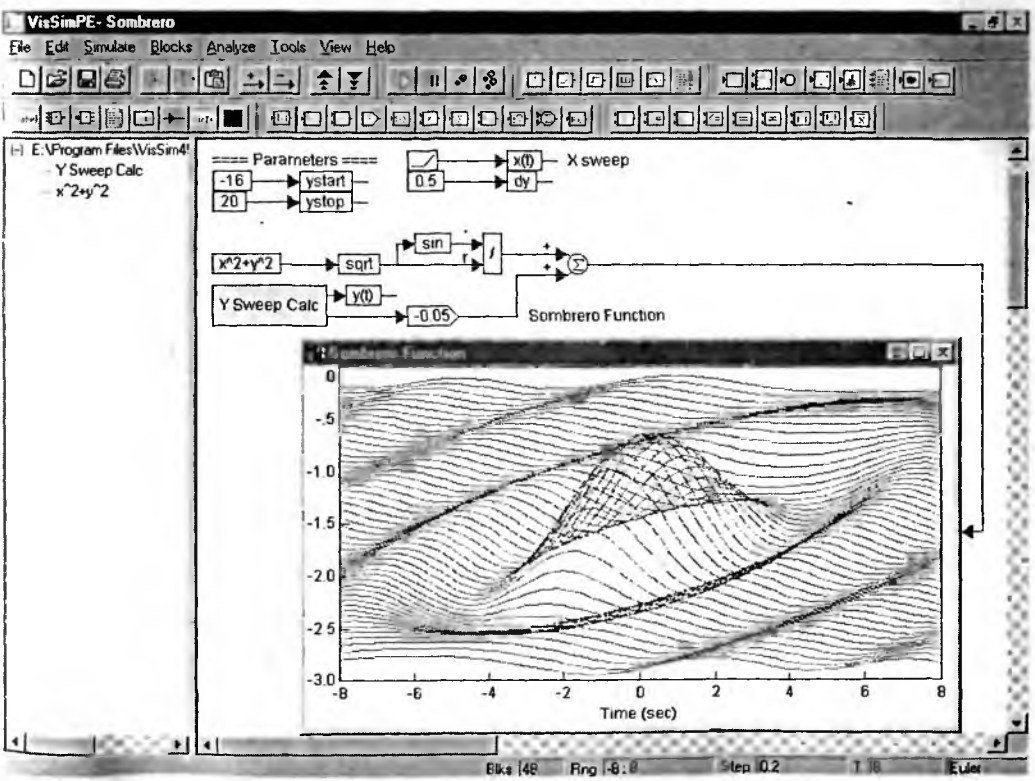


Рис. 4.68. Модель построения 3D-графика «шляпы» Sombrero

мирования аксонометрического изображения «шляпы», поскольку графопостроитель VisSim способен чертить только кривые в плоскости.

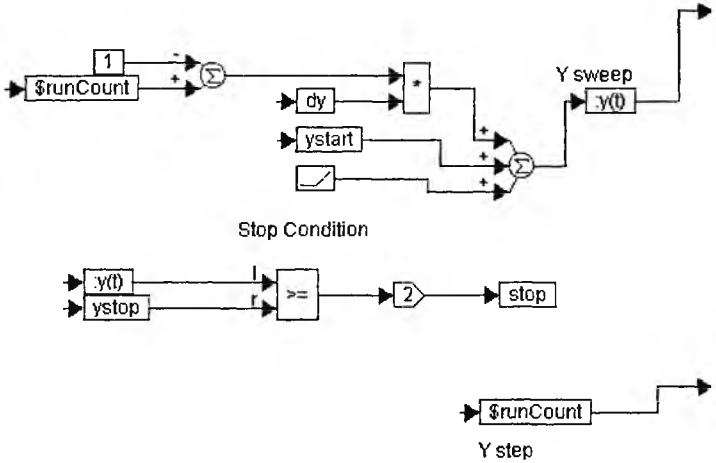


Рис. 4.69. Субблок Y Sweep Calc

Другой блок x^2 + y^2 формирует параболическую поверхность, из которой на рис. 4.70 окончательно формируется отображаемая на экране графопостроителя зависимость $z = f(x, y)$.

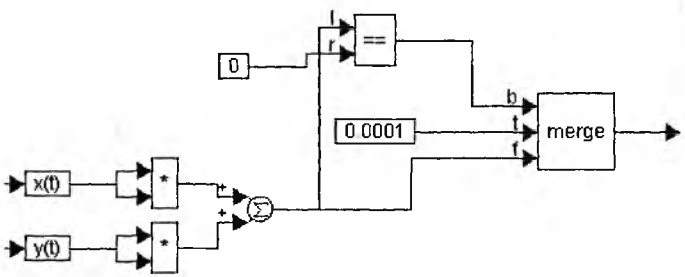


Рис. 4.70. Субблок x^2 + y^2

Алгоритм удаления невидимых линий в этом примере не предусмотрен, так что построение каркасной «шляпы» осуществляется линиями. Тем не менее фигура выглядит хотя и грубоватой, но вполне визуально понятной.

4.7.4. Вейвлет-преобразования и компрессия сигналов

В последние годы альтернативной преобразований Фурье стали вейвлет-преобразования. *Вейвлеты*, это новый базис разложения и синтеза произвольных сигналов и функций. Вейвлеты ограничены во времени, могут перемещаться по оси абсцисс или времен и масштабироваться, т. е. сжиматься или растягиваться. Большинство вейвлетов создаются итерационными мето

дами и аналитических выражений не имеют. Достаточно подробное описание вейвлетов и вейвлет-преобразований средствами СКМ можно найти в [23, 32, 37]. Там же можно найти и ссылки на основополагающие работы по вейвлетам.

Так называемые ортогональные вейвлеты могут использоваться для разложения по вейвлет-базису произвольных сигналов и функций, с последующим синтезом (реставрацией) сигналов со сколь угодно малой погрешностью. Вейвлеты гораздо лучше, чем ряды Фурье, подходят для представления и обработки сложных нестационарных сигналов, включая сигналы изображения.

В примерах на обработку сигналов в VisSim имеется пример, позволяющий познакомиться с применением вейвлет-преобразований для представления и компрессии сигналов (разработчик Allan Corbeil). Основная модель этого примера представлена на рис. 4.71.

Как видно из правой части рис. 4.71, далеко не полностью видимый список субблоков этой модели содержит десятки субблоков, что указывает на высокую сложность моделирования. Читатель может самостоятельно познакомиться с реализаций субблоков этой модели, многие из которых весьма сложны. В связи с этим ограничимся лишь общим описанием работы с данной моделью.

Модель позволяет задавать сигнал достаточно сложной формы, являющийся функцией времени и двух параметров Alpha и Beta. Их значения мож-

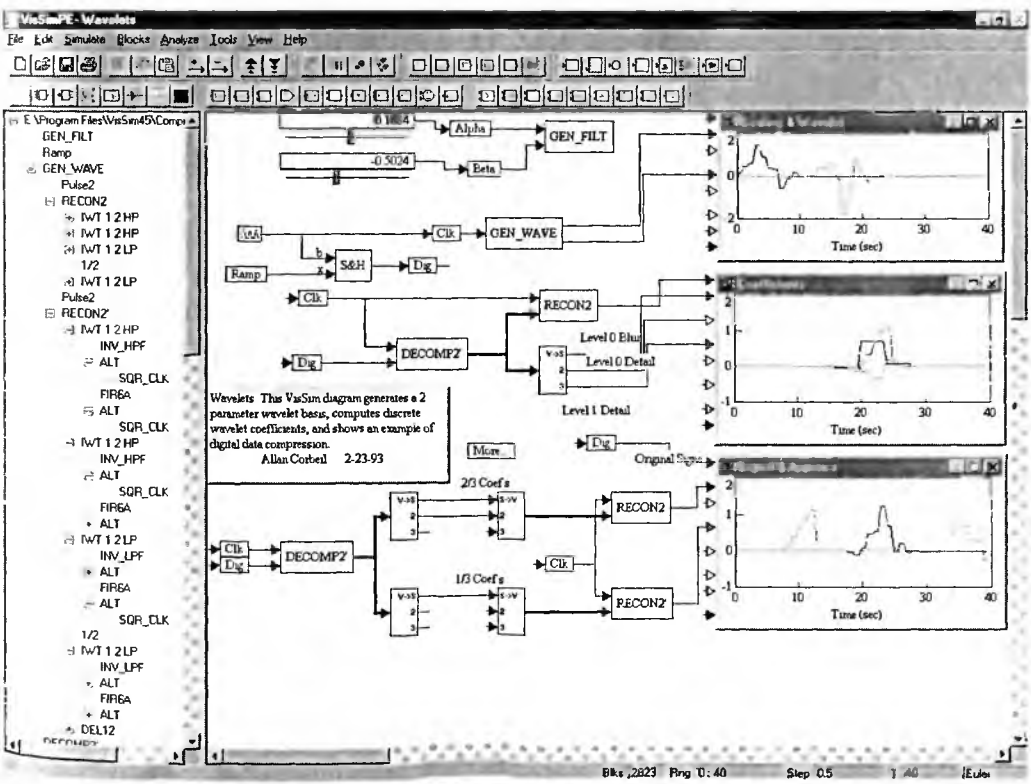


Рис. 4.71. Модель вейвлет-преобразований для представления и компрессии сигналов

но устанавливать с помощью ползунковых регуляторов, а форму сигнала наблюдать на верхнем виртуальном осциллографе. Далее модель позволяет осуществить вейвлет-анализ сигнала, т. е. его разложение по базису вейвлетов. Средний осциллограф позволяет наблюдать вейвлет-коэффициенты, которые являются специальными функциями времени. Сумма таких коэффициентов, помноженных на отсчеты сигнала, обеспечивает вейвлет-синтез сигнала, что иллюстрирует нижний осциллограф. Если число вевлет-коэффициентов, определяемое уровнем вейвлет-разложения, ограничить, то будет иметь место вейвлет-фильтрация и компрессия сигнала.

К сожалению, этот пример годится лишь для начального ознакомления с техникой вейвлет-преобразований. Практической пользы от его нет, поскольку вейвлет-преобразования средствами VisSim осуществляются слишком медленно, а сложность реализации примера показывает, что эта область моделирования для системы VisSim пока не относится к доступной в части серьезных применений. Гораздо большие возможности в этом предоставляют специальные пакеты по вейвлетам, описанные в [23, 32, 37]. Такие пакеты имеется для систем MATLAB (Wavelet Toolbox) и Mathcad (Wavelet Extension Pack).

4.8. Цифровая фильтрация сигналов

Цифровая фильтрация — одна из основных операций по обработке сигналов. VisSim имеет множество возможностей по конструированию цифровых фильтров различных типов — как непрерывных, так и дискретных.

4.8.1. Типы фильтров и методы их создания

Фильтрами называют устройства, обеспечивающие выделение из входного сигнала тех или иных временных (time domain — временной домен) или частотных (frequency domain — частотный домен) компонент. Так, издавна используемые в электросвязи и радиотехнике частотные фильтры выделяют из частотного спектра сигналов нужные частотные компоненты. Фильтры делятся на аналоговые и цифровые [39].

Аналоговые фильтры строятся на основе индуктивных и емкостных элементов и электромеханических и пьезокерамических резонаторов. Такие элементы имеют большие габариты и достаточно сложную технологию изготовления. Поэтому в последнее время широко применяются цифровые фильтры, построенные на элементах временной задержки и цифровых блоках. Такие фильтры являются основой цифровых сигнальных процессоров и выполняются по микроэлектронной технологии, обеспечивающей резкое снижение их стоимости и повышение качества фильтрации за счет применения многих звеньев.

Фактически цифровой фильтр — это дискретно-временная система, преобразующая цифровой входной сигнал в модифицированный цифровой выходной сигнал. Под это определение попадают не только фильтры, но и многие системы управления. Математически *цифровой фильтр* представляется си

стемой дифференциальных уравнений, которая может быть записана в конечно-разностном виде:

$$y(k) = \sum_{i=0}^{M} a_i x(k-i) - \sum_{j=1}^{N} b_j y(k-j).$$

Это уравнение представляет отношение между k-м отсчетом выходного сигнала и N предыдущими и M последующими значениями отсчетов входного сигнала x. Создаваемые средствами VisSim цифровые фильтры делятся на два широких класса:

1) IIR (Infinity Impulse Responce) — рекурсивные *фильтры с бесконечной импульсной характеристикой*;

2) FIF (Finite Impulse Responce) — нерекурсивные *фильтры с конечной импульсной характеристикой*, у которых все $b_j = 0$.

Фильтры могут задаваться передаточными характеристиками. К примеру, если задано уравнение дискретного фильтра класса time domain (временно́й домен)

$$y(k) = x(k) - 0.2 \cdot y(k-1) - 0.8y(k-2),$$

то фильтр может быть создан как в виде устройства с отдельными блоками, так и в виде блока передаточной функции следующего вида:

$$\frac{Y(z)}{R(Z)} = \frac{z^2}{z^2 + 0.2z + 0.8}.$$

Временные и частотные фильтры могут быть преобразованы друг в друга, что соответствует *дуальности* (двойственности) понятий времени и частоты. Для построения фильтров, в том числе тех, которые не могут быть реализованы с помощью блока передаточных характеристик, могут использоваться блоки регистров задержки 1/Z.

Наиболее распространенными являются частотные фильтры, характеризующиеся своими АЧХ и ФЧХ. В пределах каждого класса возможны разные методы построения фильтров, заданные своими наименованиями (Butterworth — Баттерворта, Bessel — Бесселя, Chebyshev — Чебышева и Inverse Chebyshev — инверсный Чебышева). Кратко отметим свойства этих фильтров на примере фильтров нижних частот.

Фильтр Бесселя обеспечивает равную временную задержку сигналов всех частот, максимально плоскую АЧХ (без колебаний в полосе пропускания и полосе задержки) и потому часто используется для частотной селекции. Этот фильтр не искажает сигнал, спектр которого лежит в пределах полосы пропускания. Но переходная характеристика этого фильтра имеет небольшое перерегулирование.

Фильтр Баттерворта имеет максимально плоскую АЧХ, что делает его предпочтительным для частотной селекции. Нормированная АЧХ такого фильтра порядка n задается аппроксимацией:

$$W(j\omega) = \frac{1}{\sqrt{\varpi^{2n-1}}},$$

где $\varpi = \omega/\omega_c$ — нормированная относительно частоты среза; ω_c — частота; n — порядок фильтра. Все производные этого фильтра от первой до $(2n-1)$-ой равны нулю.

Фильтры Чебышева (прямой и инверсный) дают максимальное подавление сигнала в области частот выше частоты среза. Они имеют колебания АЧХ, но эти колебания одинаковы по амплитуде. Наилучшая аппроксимация для таких фильтров следующая:

$$W(j\omega) = \sqrt{\frac{1}{1 + \varepsilon^2 T_n^2(\omega)}},$$

где ε — коэффициент, определяющий неравномерность АЧХ в полосе пропускания; $T_n(\omega)$ — полином Чебышева первого рода порядка n. В полосе пропускания подкоренное выражение колеблется между уровнями 1 и $1/(1 + \omega^2)$.

По виду АЧХ фильтры делятся на следующие типы:

- Low Pass — низкочастотные фильтры (ФНЧ);
- High Pass — высокочастотные фильтры (ФНЧ);
- Band Pass — полосовые пропускающие фильтры (ПФ);
- Stop Pass — полосовые заграждающие фильтры (ЗФ).

Обсуждение всех вариантов фильтров и их отличительных особенностей выходит за рамки тематики данной книги, и мы ограничимся только приведенными выше сведениями. Пользователь VisSim может за короткое время познакомиться с моделями разных фильтров и оценить их возможности с позиций своих требований.

4.8.2. Подготовка к конструированию фильтров IIR

Простейшие фильтры, например низких и высоких частот невысокого порядка, могут задаваться как блоки с соответствующими передаточными функциями. Для конструирования более сложных фильтров VisSim имеет специальный конструктор фильтров. С его применения и начнем рассмотрение процесса подготовки моделей фильтров. Зададимся целью создать фильтр низких частот Чебышева с максимально плоской АЧХ и граничной частотой 10 Гц.

Начнем с создания простейшей заготовки для модели фильтра, представленной на рис. 4.72. Она содержит генератор перепада, блок идеальной передаточной характеристики и виртуальный осциллограф. Пустив эту модель, мы увидим, естественно, на выходе единичный перепад, для удобства наблюдения сдвинутый на время 1 введением задержки в блок передаточной характеристики.

Наведя курсор мыши на блок передаточной характеристики и щелкнув правой клавишей мыши, можно вывести окно свойств передаточной функции, показанное на рис. 4.73 слева. Это окно позволяет задавать различные виды фильтров.

ФНЧ Чебышева относится к IIR-фильтрам, поэтому в окне рис. 4.73 надо активизировать кнопку IIR Filter. Это выведет окно конструктора IIR-фильтров, показанное на рис. 4.73 справа. Попутно отметим, что окно свойств передаточной характеристики позволяет задавать метод определения ее в виде коэффициентов отношения полиномов или в виде файлов с расширениями .mat или .m. Выберем вычисление коэффициентов полинома.

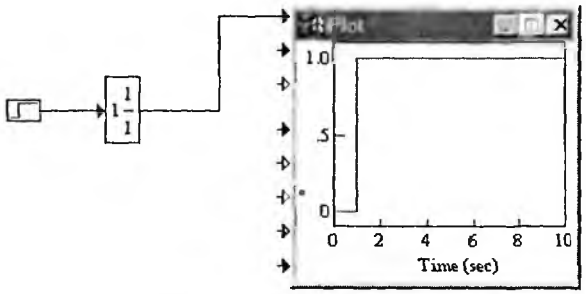


Рис. 4.72. Заготовка модели к созданию ФНЧ Чебышева

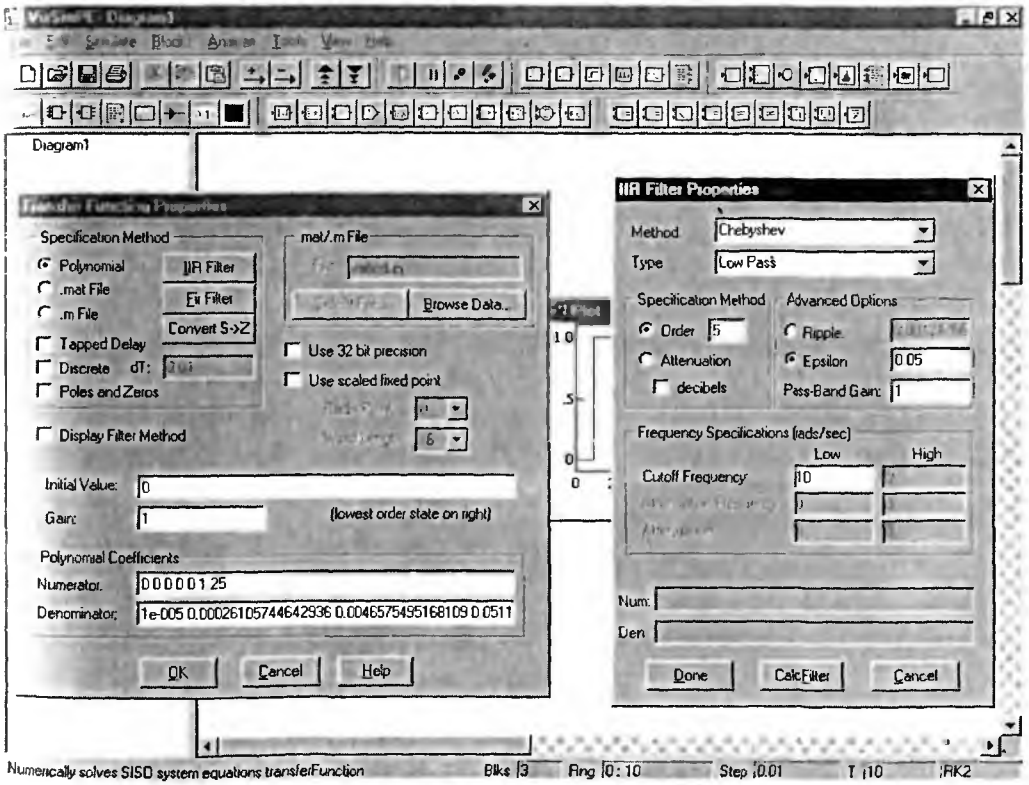


Рис. 4.73. Окна конструирования фильтров

Окно конструктора IIR-фильтров имеет два списка для задания:

1) Method — метода (имени) построения фильтров (Butterworth — Баттерворта, Bessel — Бесселя, Chebyshev — Чебышева и Inverse Chebyshev — инверсный Чебышева);

2) Type — типа фильтра по виду АЧХ.

В разделе спецификации метода Specification Method можно задать либо порядок метода, либо неравномерность АЧХ в децибелах. В разделе дополнительных опций задается волнистость Repply или погрешность Epsilon АЧХ, а также коэффициент передачи фильтра в области частот пропускания Pass

Band Gain. Выберем опцию Epsilon (фильтр с плоской АЧХ) со значением 0,05. Необходимо задать и частоты отсечки Cutt Off (две для полосовых фильтров и одну для других). В зависимости от метода построения фильтров возможны те или иные вариации приемлемых для них параметров (конструктор задает их автоматически).

Для нашего фильтра все установки представлены в окне конструктора фильтров на рис. 4.73. Для вычисления требуемых данных передаточной характеристики заданного фильтра достаточно нажать мышью кнопку Calc Filter. Коэффициенты передачи полиномов числителя (Num:) и знаменателя (Den:) появляются внизу окна. Если далее нажать кнопку Done, то эти коэффициенты появятся в окне свойств передаточной характеристики (до этого там могут быть иные данные или не быть никаких).

4.8.3. Построение переходной характеристики фильтра

Если теперь пустить модель, то будет получена переходная характеристика фильтра. Она показана на рис. 4.74. Нетрудно заметить, что переходная характеристика имеет заметные выбросы. Тут уместно вспомнить, что фильтры Чебышева гарантируют равномерные колебания АЧХ в пределах полосы пропускания или их отсутствие, но вовсе не гарантируют отсутствие выбросов у переходной характеристики.

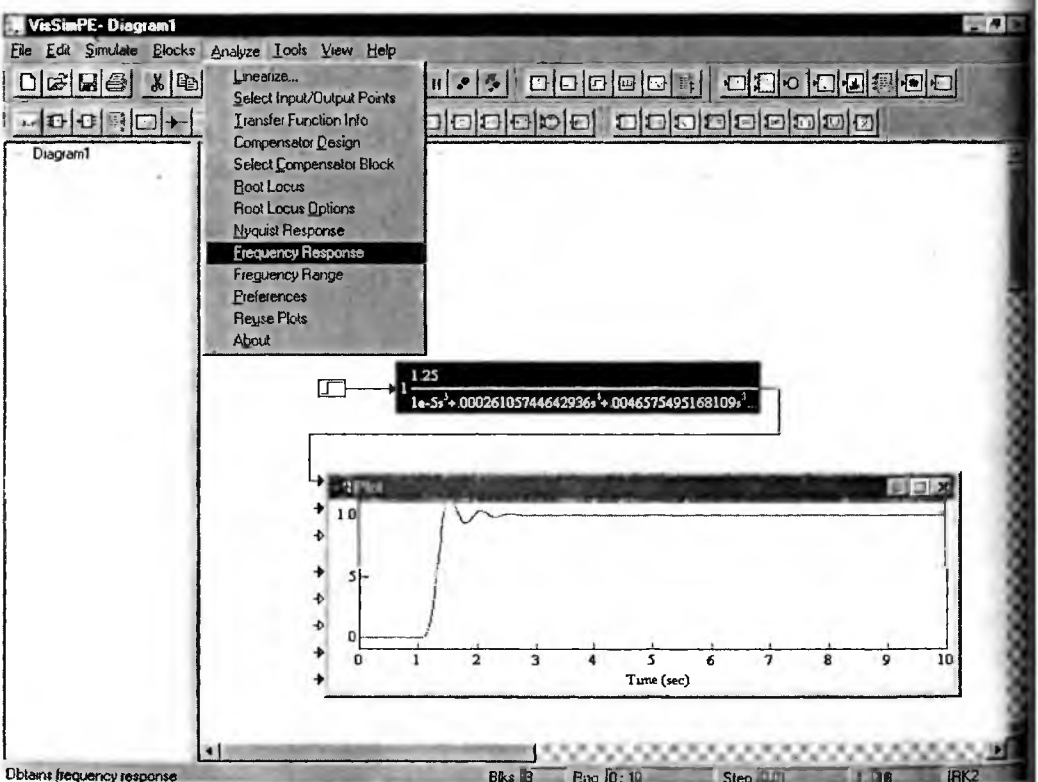


Рис. 4.74. Модель после уточнения передаточной характеристики

Полезно обратить внимание на то, что выражение в блоке задания передаточной характеристики изменилось и блок увеличился в размере. Если желательно разместить в блоке сообщение о типе созданного фильтра, то в окне свойств передаточной характеристики надо мышью проставить знак птички у опции Display Filter Method (Показать имя фильтра).

4.8.4. Построение АЧХ и ФЧХ фильтра

Теперь рассмотрим построение АЧХ и ФЧХ созданного фильтра. В принципе для этого нужно задать синусоидальный сигнал на входе фильтра с переменной частотой и задать построение графиков отношения амплитуд сигналов на входе и выходе фильтра, а также построить график разности фаз. Соответствующие блоки есть в составе VisSim, и они были описаны выше.

Однако на самом деле эти важные характеристики можно получить намного проще, поскольку с VisSim 4.5 интегрировано специальное приложение Analyze, доступ к которому возможен из одноименной позиции меню — оно показано на рис. 4.74 с открытым списком команд. Подробно работа с этим приложением будет описана в следующей главе. Здесь же мы коснемся только построения АЧХ и ФЧХ фильтров.

Все, что нужно для построения высококачественных графиков АЧХ и ФЧХ заданного фильтра, это выделить блок передаточной характеристики фильтра и исполнить в позиции Analyze команду Frequency Response (Частотный отклик). Тут же появятся окна с графиками АЧХ и ФЧХ, которые можно

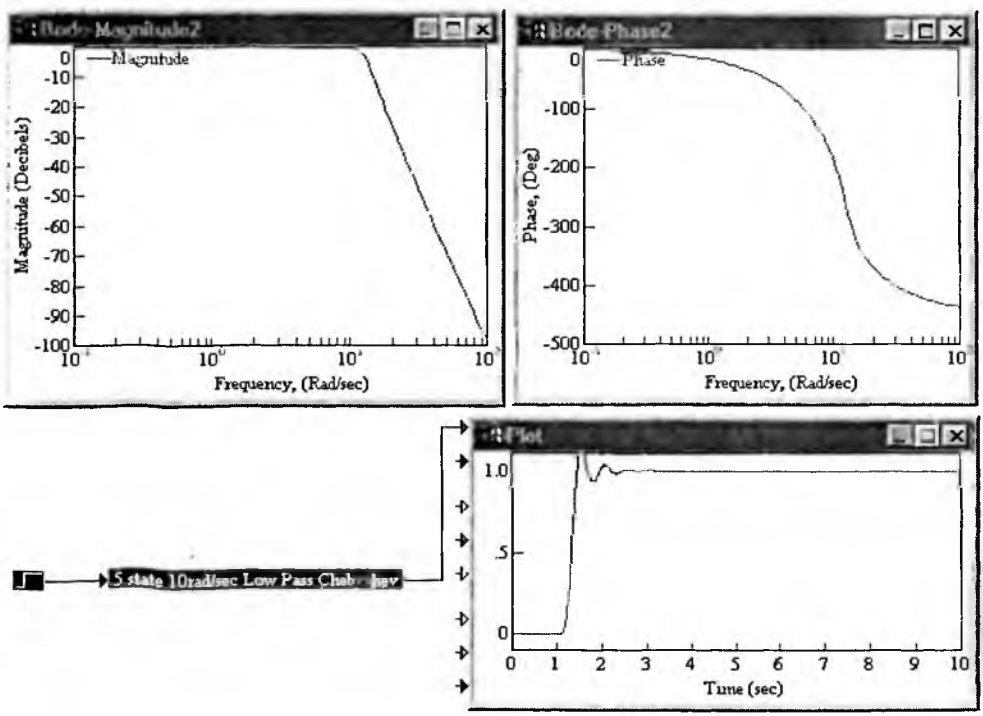


Рис. 4.75. Сконструированный ФНЧ Чебышева со всеми его характеристиками

переместить в любое место экрана и изменить в размерах. Рисунок 4.75 показывает сконструированный фильтр со всеми его характеристиками.

Как и следовало ожидать, сконструированный фильтр Чебышева имеет гладкую АЧХ, но его переходная характеристика имеет заметный выброс и даже затухающий колебательный процесс после него. В силу этого можно сделать вывод о целесообразности применения таких фильтров для частотной селекции сигналов.

4.8.5. Окно конструктора FIR-фильтров

Фильтры FIR могут быть непрерывными и дискретными. Порядок этих фильтров обычно заметно выше порядка IIR-фильтров. Окно их конструирования показано на рис. 4.76. Суть задаваемых в нем параметров вполне очевидна. В списке Kind Filter можно задать следующие типы фильтров: FIR-фильтр, дифференциатор и преобразователь Гильберта.

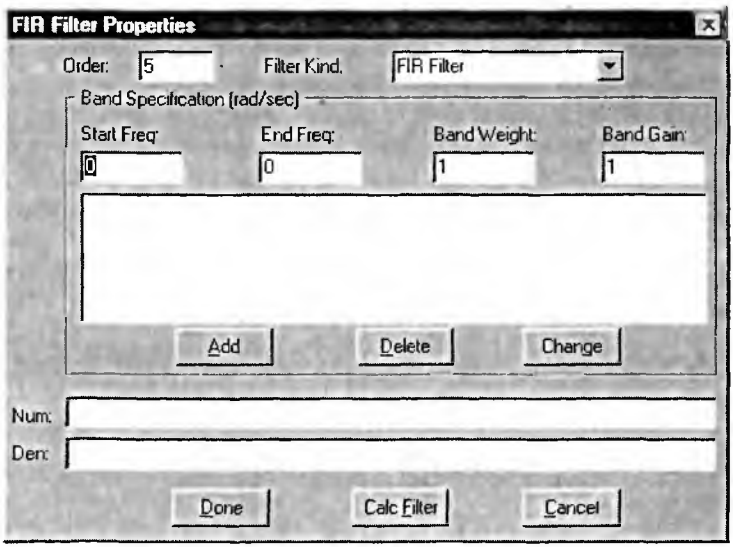


Рис. 4.76. Окно конструктора FIR-фильтров

У дискретных фильтров все задаваемые частоты должны быть ниже частоты Найквиста. У непрерывных фильтров бесконечная частота задается словом inf. Надо иметь в виду, что далеко не всякая комбинация параметров соответствует условиям реализации фильтров этого класса. Если задана недопустимая комбинация параметров, будут появляться окошки с сообщением об ошибке.

4.8.6. Пример сравнения двух фильтров

Демонстрационный пример на сравнение двух фильтров дан на рис. 4.77. Здесь представлено построение двух типов фильтров — пятизвенного низкочастотного фильтра Баттерворта и двухполосного десятизвенного фильтра с ограниченной импульсной характеристикой FIR.

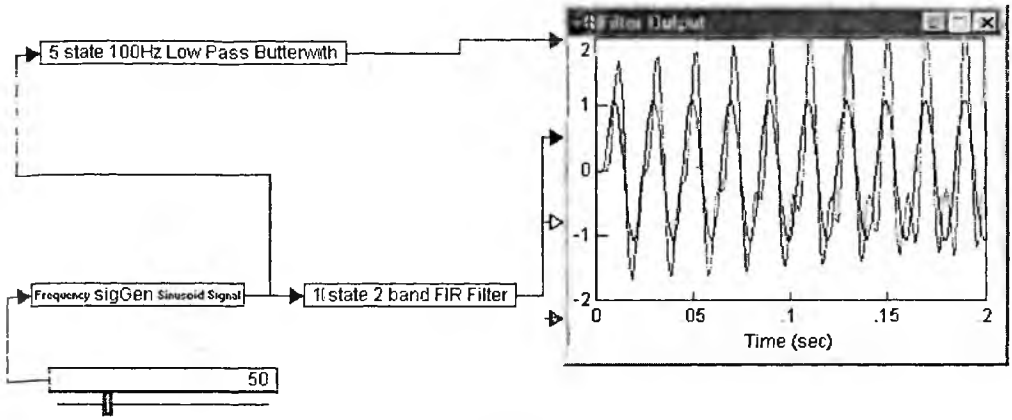


Рис. 4.77. Примеры задания фильтров с помощью блоков transferFunction

Оба фильтра задаются как блоки типа transferFunction. Щелкнув правой кнопкой мыши по такому блоку, можно открыть окно свойств блока и просмотреть (а при необходимости и скорректировать) параметры фильтров.

Для задания синусоидального тестового сигнала вида $\sin(2\pi ft)$ служит блок SigGen (рис. 4.78), частота которого задается ползунковым регулятором и отображается цифровым индикатором. Это позволяет легко и наглядно оценить прохождение синусоидального сигнала через заданные фильтры.

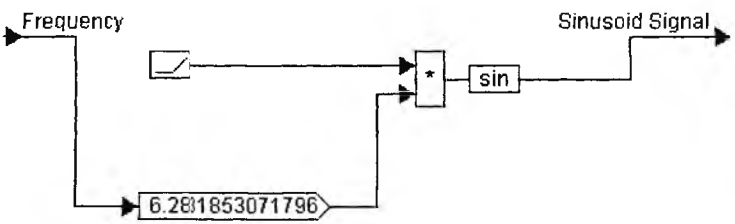


Рис. 4.78.Субблок задания синусоидального сигнала

Стоит отметить, что осциллограммы фильтров фактически регистрируют их реакцию на появление синусоидального сигнала заданной частоты с момента времени $t = 0$. Таким образом оценивается переходной процесс на выходе фильтра при подаче ступеньки синусоидального воздействия.

4.6.7. Пример реализации фильтра на основе вычислений с фиксированной точкой

При построении цифровых фильтров остро стоит проблема повышения их эффективности и увеличения рабочих частот. Один из путей решения этой проблемы заключается в использовании вычислений с фиксированной точкой, которые реализуются на аппаратном уровне. Пример такого фильтра за-

дан в файле Fixptfil.vsm, расположенном в папке Fixptdsp. Основная модель этого примера показана на рис. 4.79.

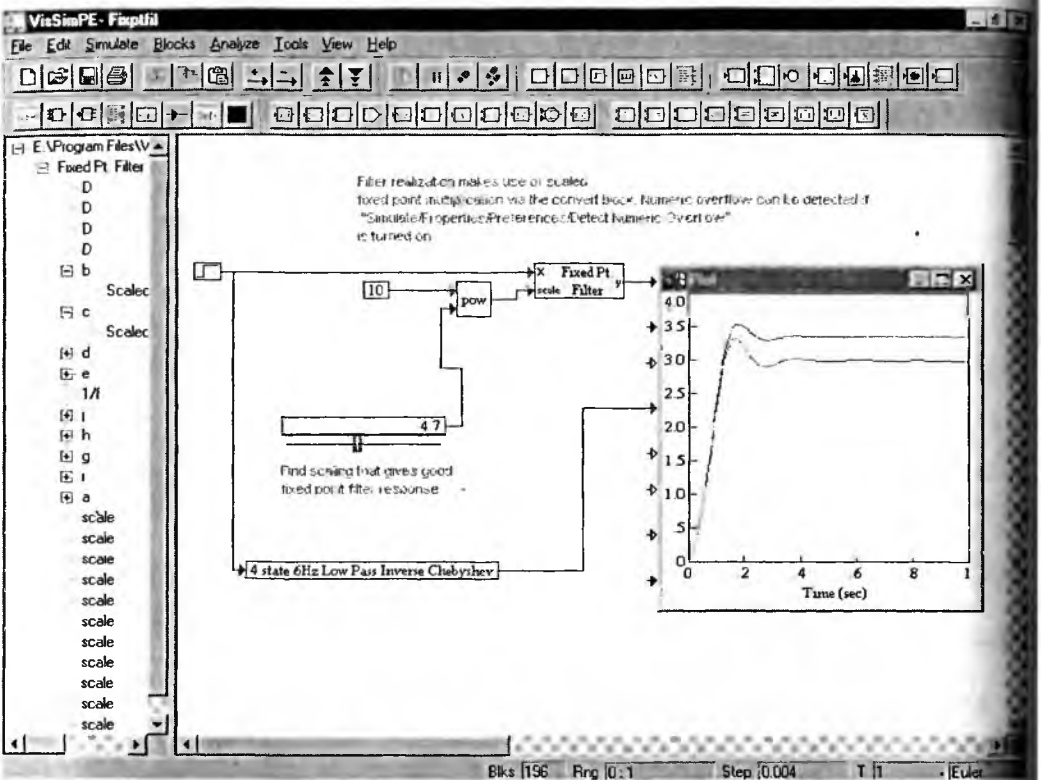


Рис. 4.79. Основная модель примера Fixptfil

В этом примере сравниваются две реализации ФНЧ Чебышева 4-го порядка — на основе блоков с вычислениями в формате чисел с фиксированной точкой и на основе передаточной характеристики. Производится подстройка первого фильтра с помощью ползункового регулятора, что позволяет подстраивать переходную характеристику этого фильтра. При параметре подстройки 4.7 переходные характеристики сравниваемых фильтров немного различаются, что и видно на рис. 4.79. При значении параметра 4.8 характеристики фильтров практически совпадают.

Как видно из левой части рис. 4.79, реализация фильтра на основе блоков с вычислениями в формате чисел с фиксированной точкой довольно сложна и содержит многие десятки в основном однотипных блоков (рис. 4.80).

С субблоками более низкого уровня можно ознакомиться, активизируя их мышью. Реализация данного фильтра может показаться чрезмерно сложной, но не надо забывать, что это цифровой фильтр и его блоки легко реализуются в составе интегральных микросхем. В этом случае количество блоков играет далеко не главную роль. Гораздо важнее простота и миниатюрность их реализации.

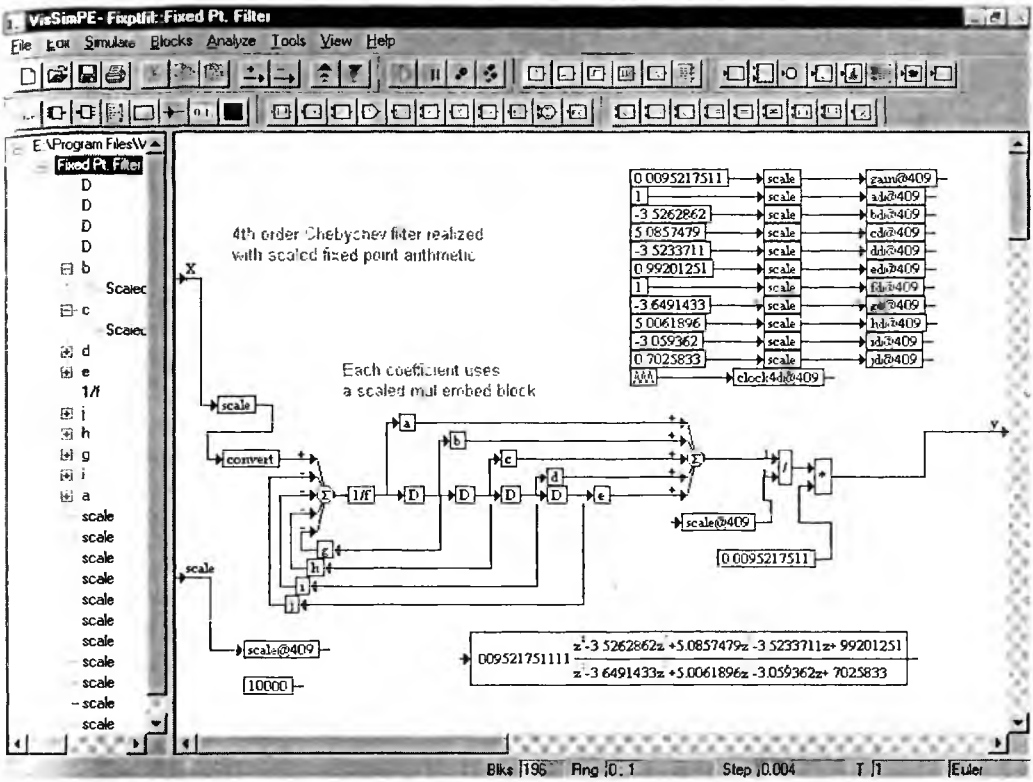


Рис. 4.80. Субблок первого уровня реализации фильтра на основе блоков с вычислениями в формате чисел с фиксированной точкой

В числе демонстрационных примеров VisSim можно найти реализацию фильтра Калмана. К сожалению, пример (в используемой автором версии VisSim 4.5) не работоспособен, а потому в данной книге эта реализация фильтра не рассматривается.

Глава 5
Математическое моделирование систем

Моделирование систем самого различного назначения — одна из наиболее перспективных областей применения системы моделирования VisSim. Этому способствует математическая ориентация многих блоков VisSim. В этой главе описываются практические примеры моделирования и исследования различных систем с применением для этого средств системы VisSim.

5.1. Линейные и нелинейные системы

5.1.1. Начальные сведения о системах

В этой главе рассматриваются сложные составные объекты — *системы*, состоящие из множества связанных между собой элементов. Элементом является неделимый в рамках данной системы компонент сложных объектов, явлений и процессов. Элемент может в свою очередь быть системой более низкого уровня или подсистемой. Систему можно также определить и как множество с некоторыми дополнительными характеристиками.

Многие даже простые технические устройства являются системами, поскольку в них явно выделяется множество отдельных элементов и даже входящих в них подсистем. Если такие системы используются для управления и контроля, то их называют *управляющими системами* или *системами контроля*.

Работа и характеристики систем зависят от свойств элементов систем. Например, для получения электронного усилителя с большим коэффициентом усиления мы вынуждены строить усилители с большим числом каскадов. Однако, какие бы каскады мы не применяли, усилитель не может превратиться в генератор синусоидальных колебаний. Чтобы это произошло, надо охватить усилитель положительной или, строго говоря, комплексной *обратной связью*. Под ней понимается подача всего выходного сигнала или его части на вход системы, где эта часть выходного сигнала суммируется со входным сигналом. А если мы хотим получить усилитель с высоким качеством (малыми частотными и нелинейными искажениями, стабильным коэффициентом усиления и т. д.), то придется применить в усилителе отрицательную обратную связь, при которой сигнал обратной связи вычитается из входного сигнала.

Таким образом, наличие обратных связей является характерным признаком многих технических систем. Более того, объединяя однотипные элементы в одно целое, нередко можно свести техническую систему к трем компонентам — сумматору, прямому каналу и каналу обратной связи.

Если коэффициенты передачи элементов систем не зависят от уровня сигналов, то системы относятся к *линейным системам*. В противном случае мы имеет *нелинейные системы*. Могут быть системы с неизменной структурой и с переменной структурой. Изменение структуры систем достигается включением в них логических блоков, включающих или выключающих отдельные компоненты систем. У *параметрических систем* некоторые параметры зависят от времени.

5.1.2. Линейные системы с комплексной обратной связью

Рассмотрим простейшие линейные системы. К таким системам можно отнести усилители, работающие при малых уровнях сигнала, например операционные. На рис. 5.1 представлены три варианта моделей простейших линейных устройств.

Вариант, представленный сверху, соответствует представлению о классическом усилителе, охваченном отрицательной обратной связью. Для ее осуществления в усилителе используется суммирующее устройство, нижний вход которого задается как вычитающий. В системе отчетливо выделяется прямой канал и канал обратной связи, через который часть выходного сигнала подается на инвертирующий вход усилителя.

В общем случае, как сама система, так и канал обратной связи имеют комплексные коэффициенты передачи. При этом они представляются дейст-

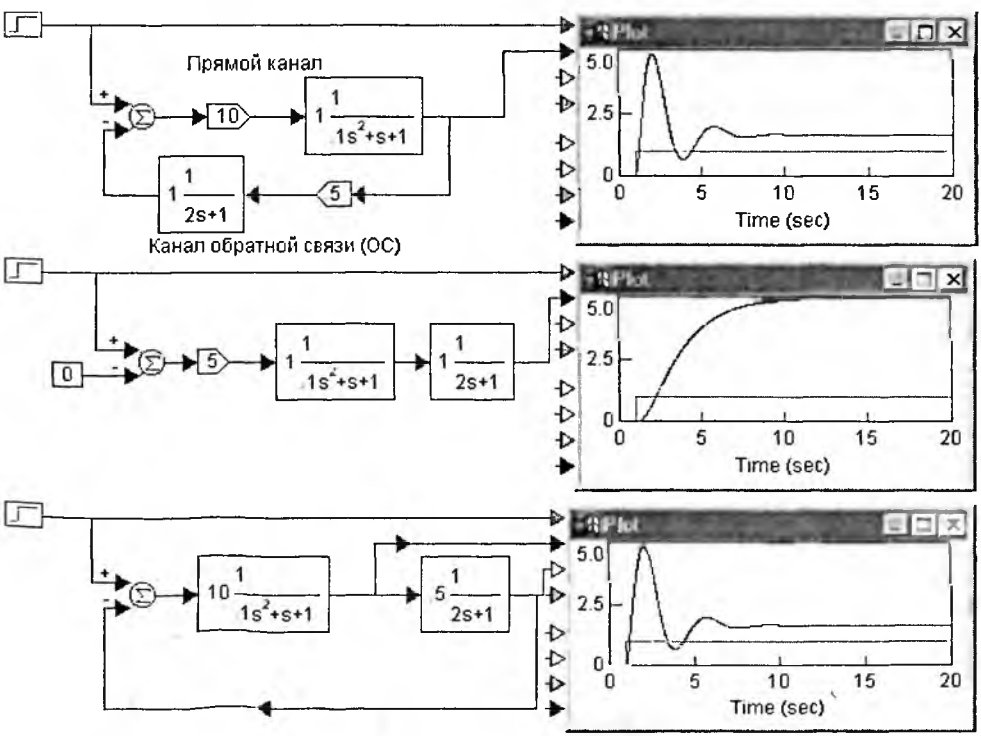


Рис. 5.1. Три варианта простых линейных систем

вительными коэффициентами передачи, умноженными на соответствующие передаточные функции. В нашем случае действительный коэффициент передачи основного модуля системы равен 10, а действительный коэффициент передачи цепи обратной связи равен 0,5.

Если бы коэффициенты передачи были только действительными величинами, то переходная характеристика такой системы была бы ступенькой. Но на самом деле она в нашем случае имеет достаточно сложный вид, что и показывают осциллограммы реакции системы на единичный скачок. Хорошо видно, что выходной сигнал содержит явно заметную колебательную затухающую компоненту. Все это указывает на то, что на самом деле обратная связь в данном примере является комплексной. На каких-то частотах она может иметь действительную составляющую, но на других еще и мнимую составляющую.

Из теории обратной связи [34, 40] известно, что комплексный коэффициент передачи системы с обратной связью, осуществляемой через инвертирующий вход, выражается как:

$$\dot{K}(\omega) = \frac{\dot{K}_y(\omega)}{1 + \dot{\beta}(\omega) \cdot \dot{K}_y(\omega)}$$

или в операторной форме

$$K(p) = \frac{K_y(p)}{1 + \beta(p) \cdot K_y(p)}.$$

Если пренебречь частотной зависимостью коэффициентов передачи прямого канала системы (усилителя) K_y и канала обратной связи β, то можно получить следующее выражение для коэффициента передачи безынерционной системы (усилителя) с отрицательной обратной связью:

$$K = \frac{K_y}{1 + \beta \cdot K_y}.$$

В нашем случае $K_y = 10$, а $\beta = 0{,}5$, таким образом $K = 10/(1 + 0.5 \cdot 10) = 10/6 \approx 1.667$. Это соответствует установившемуся значению сигнала на выходе нашей системы (рис. 5.1 сверху). Заметим, что если бы обратная связь осуществлялась через неинвертирующий вход, то знак + в знаменателе последнего выражения надо было бы заменить на –. При этом обратная связь при отсутствии частотных искажений классифицируется как положительная. Пример с такой связью мы рассмотрим чуть ниже.

В природе идеальных (безынерционных) систем не бывает. Если в каналах системы есть инерционные элементы, то коэффициенты передачи становятся комплексными, и это можно учесть соответствующими передаточными характеристиками в операторной форме, что и сделано в нашем случае заданием передаточных характеристик. В результате вычисление комплексного коэффициента передачи системы и ее переходной характеристики заметно усложняется, но не в случае применения системы VisSim. Как нетрудно видеть из представленного на рис. 5.1 примера, VisSim легко моделирует реакцию инерционной системы на единичный скачок, т. е., по существу, рассчитывает ее переходную характеристику.

5.1.3. Система с разомкнутой обратной связью

Для теоретического анализа систем с обратной связью важное значение имеет случай, когда цепь обратной связи разомкнута. В этом случае «кольцо» системы разрывается и выпрямляется. Мы получаем систему, которая просто состоит из двух включенных последовательно звеньев. Этот случай показан на рис. 5.1 в центре.

Тут интересно отметить, что наша система при разрыве цепи обратной связи имеет монотонную переходную характеристику. Это и является главным достоинством систем с разорванной обратной связью — их можно исследовать, не опасаясь нестабильности, могущей возникнуть при введении обратной связи. Разумеется, речь идет об идеализации модели, т. е. исключении паразитных обратных связей. Электронщики прекрасно знают, что усилитель с очень большим коэффициентом усиления может самовозбудиться даже при отсутствии специальных цепей обратной связи из-за наличия паразитных цепей.

И в самом деле, при наличии «отрицательной» обратной связи она имеет колебательную переходную характеристику. И более того, может привести к возникновению не только затухающих, но и нарастающих колебаний.

Комплексный коэффициент передачи системы с разорванной обратной связью определяется выражением $\dot{K}(\omega)\cdot\dot{\beta}(\omega)$, т. е. определяется произведением комплексного коэффициента передачи системы (усилителя) на комплексный коэффициент передачи звена обратной связи.

5.1.4. Простейшая линейная система управления

В теории автоматического управления и регулирования принято объединять компоненты линейных систем в одну цепочку [40]. Такой подход реализован в третьем варианте построения системы, представленном на рис. 5.1 снизу. Этот вариант можно рассматривать как предыдущий, но теперь уже замкнутый в кольцо, т. е. с восстановленной цепью отрицательной обратной связи.

При внимательном рассмотрении последнего варианта модели системы нетрудно заметить, что блоки «усиления» gain исключены. Но зато в блоках передаточных характеристик проставлены параметры Gain = 10 и 0,5. Это позволяет несколько упростить модель системы, сохранив все ее характеристики. Как нетрудно заметить из осциллограмм, передаточные характеристики первого и третьего вариантов построения простой линейной системы абсолютно идентичны. Это следствие идентичности самих систем.

5.1.5. Нестабильные линейные системы

Итак, мы установили, что комплексная обратная связь при определенных передаточных характеристиках и в определенном диапазоне частот может вести себя как действительная обратная связь. Последняя обычно подразделяется на отрицательную и положительную. При отрицательной обратной связи сиг-

нал с выхода подается на вход в противофазе, что ведет к уменьшению общего коэффициента передачи системы. При положительной обратной связи сигнал с выхода подается в фазе со входным сигналов, что увеличивает коэффициент передачи, равный:

$$K = \frac{K_y}{1 - \beta \cdot K_y}.$$

Однако если $\beta \cdot K_y \to 1$, то коэффициент передачи системы $K \to \infty$. На практике это означает нестабильность системы — выходной сигнал периодически возрастает до неопределенно больших значений.

В случае комплексной обратной связи есть различные критерии нестабильности. Например, принято считать, что если на какой-то частоте ω_0 суммарный фазовый сдвиг в системе равен 0, а коэффициент передачи системы больше или равен 1, то на этой частоте в системе возникают и нарастают синусоидальные колебания. Из этого условия вытекают и способы графического анализа устойчивости, например с применением диаграмм Найквиста (годографов) и Николса.

Рисунок 5.2 демонстрирует модели линейных систем с этими видами нестабильности. В первом случае выходной сигнал стремится к очень большим значениям, что может даже вызвать переполнение разрядной сетки чисел у компьютера. Во втором случае видны экспоненциально нарастающие синусоидальные колебания. На самом деле такое нарастание амплитуды колебаний рано или поздно приводит к возникновению нелинейных эффектов.

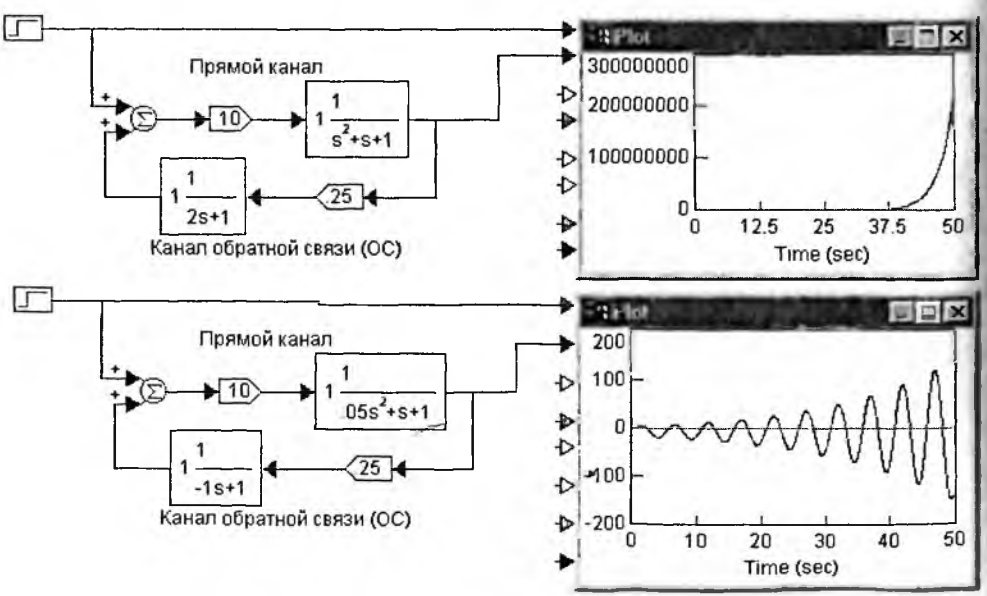


Рис. 5.2. Примеры нестабильных линейных систем

5.1.6. Нелинейные системы

Анализ нелинейных систем — задача намного более трудная, чем анализ линейных систем. Более того, в аналитическом виде она решена лишь для небольшого числа частных случаев. И только моделирование численными методами способно дать наглядное представление о режимах работы и параметрах нелинейных систем.

Рассмотрим две простые нелинейные системы, представленные на рис. 5.3. Первая — это уже описанная ранее система, дополненная блоком нелинейности с порогами ограничения ±10. Поведение системы довольно необычно — вначале развиваются нарастающие колебания, но затем система вдруг залипает в стабильном состоянии с отрицательным уровнем сигнала.

Назвать причину этого явления с первого взгляда трудно. Между тем она заключается в специфике выбора передаточных характеристик. Оба примененных звена пропускают постоянную составляющую сигнала. И если петлевое усиление системы выше 1 (что и имеет место), то нелинейная система будет иметь два устойчивых состояниями равновесия (подобно триггеру в электронике). После некоторых колебаний она и попадает в одно из таких состояний. В нем дифференциальный коэффициент передачи прямого нелинейного канала падает до 0, и это состояние оказывается стабильным.

Во втором случае в системе использовано единственное инерционное звено, передаточная характеристика которого имеет пик на некоторой частоте и затем падает вплоть до 0 по обе стороны от пика. В итоге функциями триггера

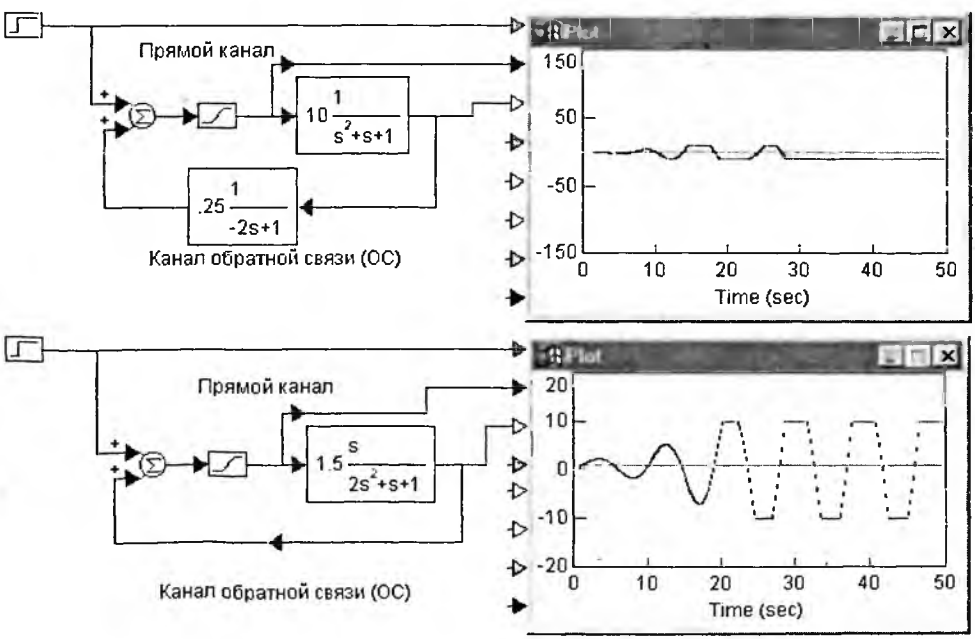


Рис. 5.3. Примеры нелинейных систем с разными видами передаточных характеристик

система не обладает и может выполнять функции автогенератора стабильных по амплитуде почти синусоидальных колебаний. На это и показывают осциллограммы выходных сигналов этой системы.

Рассмотрим еще одну нелинейную систему и сравним ее с подобной линейной системой (см. рис. 5.4). В этих системах прямой канал не пропускает постоянную составляющую сигнала, а обратный пропускает. В такой линейной системе колебания непрерывно нарастают по амплитуде, а в нелинейной после начального роста они быстро стабилизируются по амплитуде. Таким образом, вторая система может служить на практике генератором незатухающих колебаний.

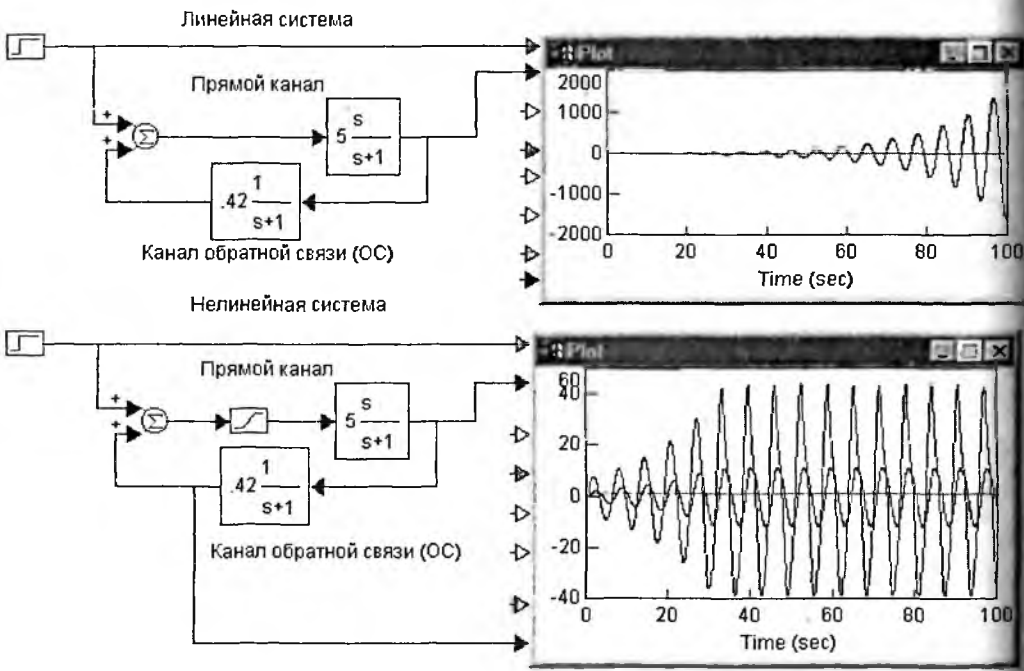


Рис. 5.4. Сравнение линейной и нелинейной систем с разными передаточными характеристиками

В задачи данного раздела входит лишь демонстрация того, насколько эффективно можно применять VisSim для моделирования линейных и нелинейных систем. Разумеется, рассмотрение всего многообразия видов таких систем и режимов их работы выходит за рамки данной книги. Мы остановимся на моделировании наиболее распространенных систем.

5.1.7. Системы под внешним воздействием

Теперь пора отметить, что нелинейные колебательные системы могут быть автономными и неавтономными. *Автономные* системы генерируют колебания сами по себе без каких-либо внешних воздействий. *Неавтономными* называют системы, находящиеся под внешним воздействием.

Показанные на рис. 5.3 и 5.4 нелинейные системы относятся к неавтономным системам, поскольку на их входе действует сигнал в виде перепада. Он задает системе начальный толчок, после чего колебания в ней развиваются сами по себе. Рисунок 5.5 иллюстрирует моделирование реальных неавтономных систем, у которых колебания возникают самопроизвольно, но благодаря броску сигнала в первом случае (модель сверху) и благодаря имитации шумов включением на вход источника шума во втором случае. В обоих случаях по мере нарастания сигнала на выходе роль возмущающих факторов резко снижается и установившиеся колебания имеют параметры, не зависящие от них. Вы можете убедиться сами в том, что вид шума (например, шум с равномерным или нормальным распределением вероятности) не влияет существенно на вид установившихся колебаний, но может повлиять на характер начального роста амплитуды колебаний.

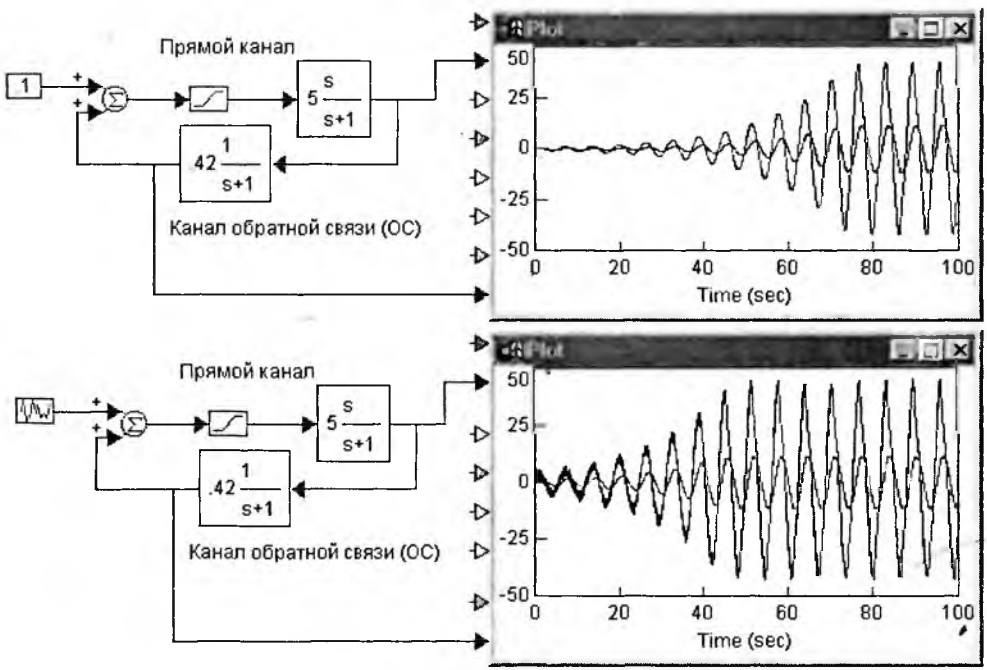


Рис. 5.5. Моделирование реальных автономных систем

В качестве дополнительных примеров на рис. 5.6 показаны линейная и нелинейная системы, на входе которых действует синусоидальный сигнал с амплитудой 1 и частотой 1,5. На линейную систему сигнал влияет мало, поскольку амплитуда колебаний в ней быстро становится очень большой, так что входной сигнал подавляется. А вот в нелинейной системе отчетливо видны биения сигнала даже при выходе на стационарный режим. Имеет место и явление *синхронизации*, при котором частоты входного и выходного сигналов находятся в строго кратном отношении.

Приведенные примеры говорят о том, что одних моделей и выражений для передаточных характеристик их компонентов недостаточно, чтобы легко

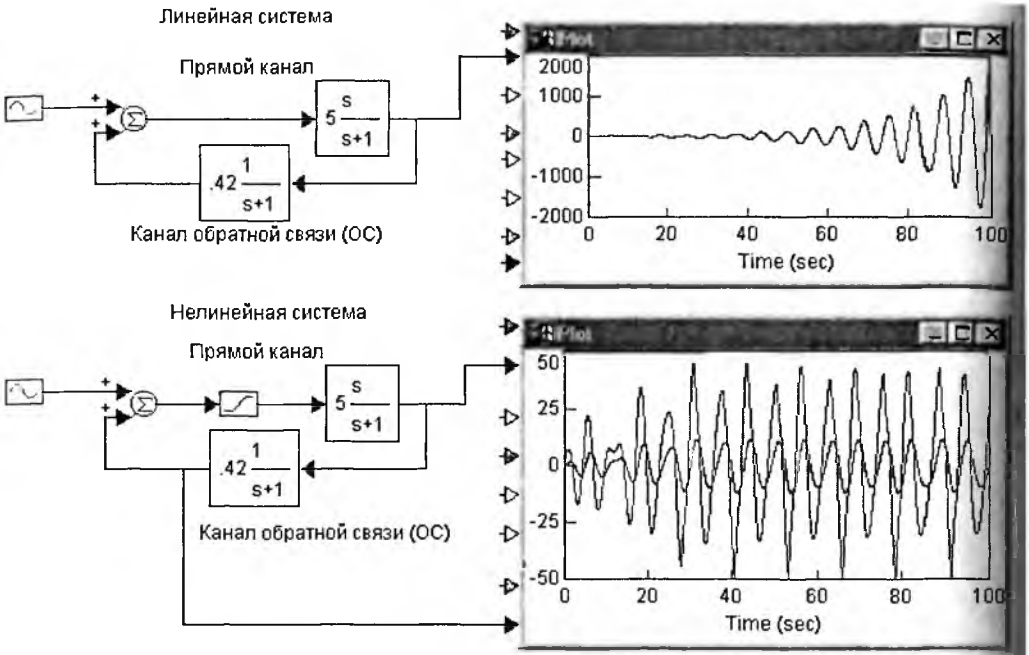


Рис. 5.6. Модели неавтономных систем — линейной и нелинейной

судить о свойствах и поведении даже простых систем. Нужны инструменты для более тщательного исследования систем, в частности построения АЧХ и ФЧХ систем в целом и их компонентов, построения диаграмм устойчивости, отображения нулей и полюсов передаточных характеристик и т. д. К их описанию мы и перейдем.

5.2. Приложение Analyze для оперативного анализа

5.2.1. Позиция Analyze меню VisSim

В состав VisSim вошло новое средство для оперативного анализа систем. Это анализатор систем Analyze, который создает в меню свое подменю с таким же названием (рис. 5.7). Для применения этого средства достаточно выделить блоки анализируемой подсистемы, даже если она входит в общую систему. После этого исполняется нужная команда, и тут же получаются окна с результатами анализа.

В позиции Analyze имеется следующий набор команд:

- Linearize... — вывод окна с данными о линеаризованной модели;
- Select Input/Output Point — селекция входов и выходов;
- Transfer Function Info — вывод окна с информацией о передаточной функции подсистемы;
- Compensator Design — конструирование компенсатора;
- Select Compensator Block — выделение блока компенсатора;

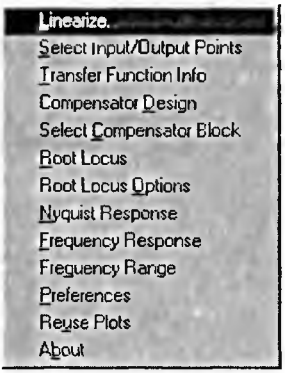


Рис. 5.7. Подменю позиции Analyze меню VisSim

- Root Locus — вывод диаграммы нулей и полюсов;
- Root Locus Options — установка опций нулей и полюсов;
- Nyquist Responce — построение диаграммы Найквиста;
- Frequence Responce — построение АЧХ и ФЧХ в отдельных окнах;
- Frequence Range — установка пределов шкалы частот;
- Preferences — вывод окна установок Analyze;
- Reuse Plot — построение нескольких графиков;
- About — вывод окна с данными о версии расширения Analyze.

5.2.2. Получение линейной модели системы

Рассмотрим в качестве примера простую модель усилителя с обратной связью (рис. 5.8). В целом модель представляет собой усилитель, подключенный к источнику перепада и к осциллографу. Это позволяет сразу получить переходную характеристику усилителя вместе с временной зависимостью сигнала (с множителем 100 для выравнивания масштабов входного и выходного сигналов).

Здесь уместно отметить, что VisSim позволяет получать большие и четкие графики любых характеристик, в том числе и переходной. Для этого достаточ-

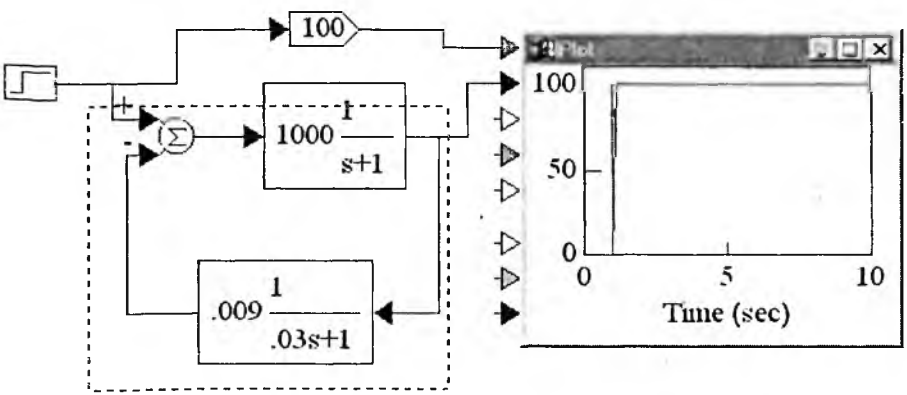


Рис. 5.8. Пример модели усилителя

но открыть соответствующее окно графика. Такое окно для переходной характеристики показано на рис. 5.9.

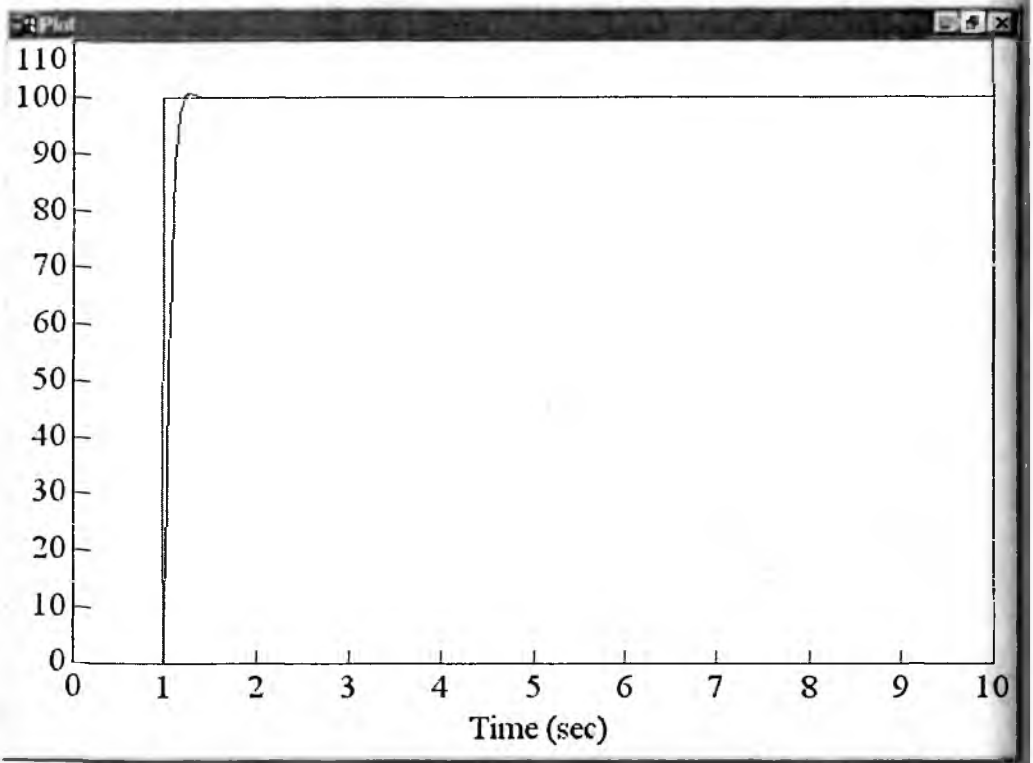


Рис. 5.9. Полностью открытое окно с переходной характеристикой усилителя

С помощью мыши выделим усилитель с цепью обратной связи. Выделение на рис. 5.10 инверсией цвета фона блоков. Затем необходимо выделить входы/выходы системы, которые будет использоваться для анализа системы. Для этого надо исполнить команду Select Input/Output Point. Курсор мыши превратится в крестик и им надо указать входы/выходы. Если это сделано правильно, то цвет входов/выходов меняется, иначе VisSim издает предупреждающий сигнал. Выделить входы/выходы можно и позже. В нашем примере надо выделить выход источника входного сигнала (это вход подсистемы) и второй сверху вход осциллографа (это выход подсистемы).

Теперь можно исполнить команду Linearize... для получения информации о линейной модели системы (заметим, что система при этом может быть как линейной, так и нелинейной). Появится окно System Linearization, показанное на рис. 5.10 справа. В этом окне надо задать имя файла, который нужен для этой операции. Если нужные входы/выходы не были помечены, то это можно сделать, нажав кнопку Select Input/Output Point. Установите также опции вывода Output Selection данных линеаризированной модели в m-файл или на экран дисплея (Linearize to .m File или Linearize to Screen Display).

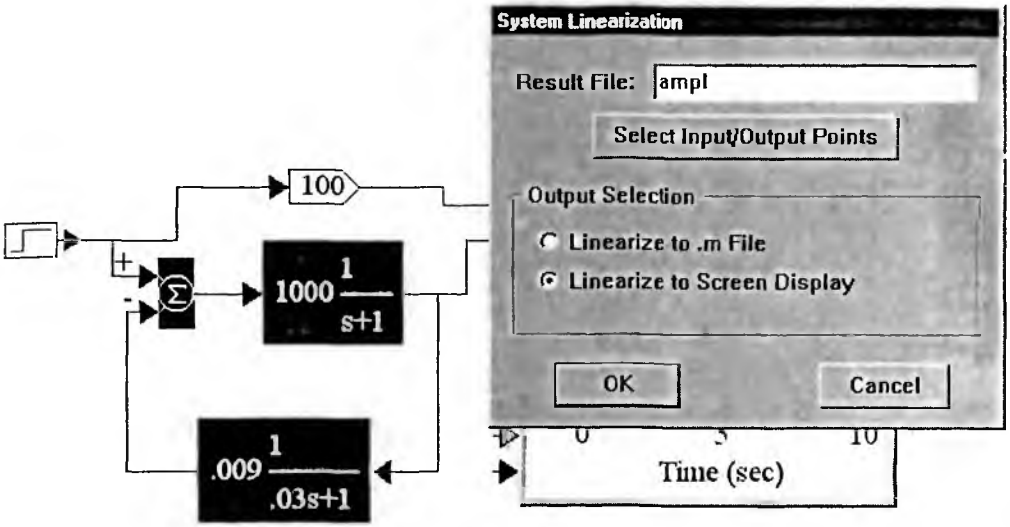


Рис. 5.10. Выделение анализируемой подсистемы и подготовка к ее анализу

Задав вывод данных на экран дисплея, нажмем клавишу OK. Начнется вывод данных о линейной модели в пространстве состояний. Выводимые окна представлены на рис. 5.11—5.14.

Эти четыре матрицы полностью определяют модель системы в терминах уравнений состояния.

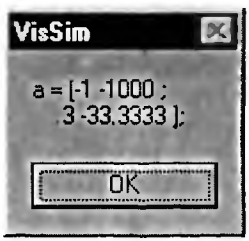


Рис. 5.11. Окно вывода матрицы a

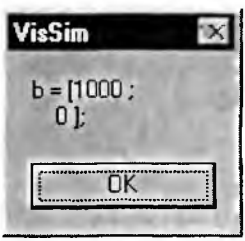


Рис. 5.12. Окно вывода матрицы b

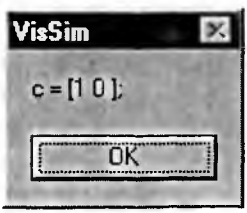


Рис. 5.13. Окно вывода матрицы c

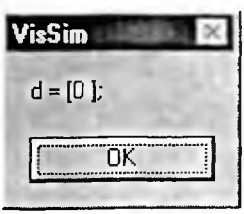


Рис. 5.14. Окно вывода матрицы d

5.2.3. Получение передаточной функции системы

Для получения передаточной функции системы (точнее, выделенной подсистемы) надо исполнить команду Transfer Function Info. Появится окно с данными передаточной функции, показанное на рис. 5.15. В этом окне данные передаточной функции относятся к системе без обратной связи, поскольку последняя размыкается.

Передаточная функция задастся отношением полиномов числителя и знаменателя, коэффициенты которых и приведены в окне рис. 5.15. Если создать блок передаточной характеристики с этими данными, то его переходная характеристика должна быть такой же, как у выделенной подсистемы. Это и показывает рис. 5.16.

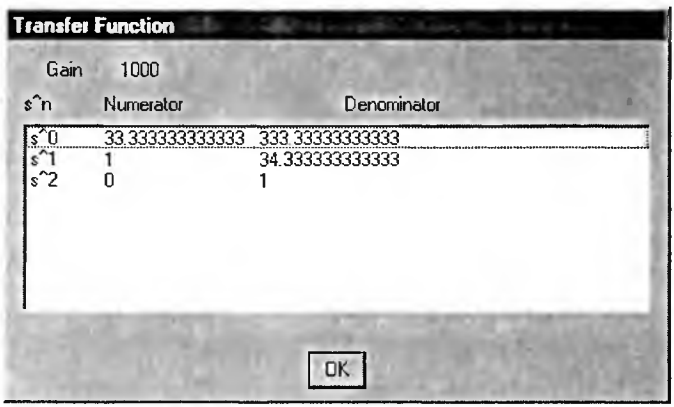


Рис. 5.15. Окно с данными передаточной функции

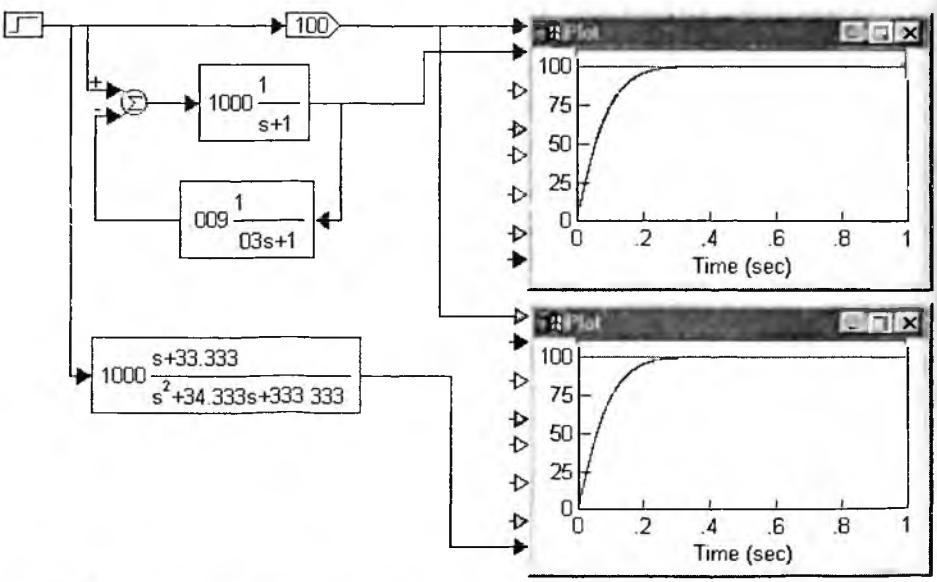


Рис. 5.16. Исходная система и эквивалентный ей блок передаточной характеристики

Переходные характеристики системы и блока передаточной характеристики под ней практически идентичны, что указывает на правильность вычисления передаточной характеристики данной системы (усилителя) с обратной связью.

5.2.4. Информация о нулях и полюсах передаточной функции

Но продолжим экспресс-анализ нашей подсистемы. Если закрыть окно рис. 5.15, то появится новое окно с данными о нулях и полюсах передаточной функции. Оно показано на рис. 5.17.

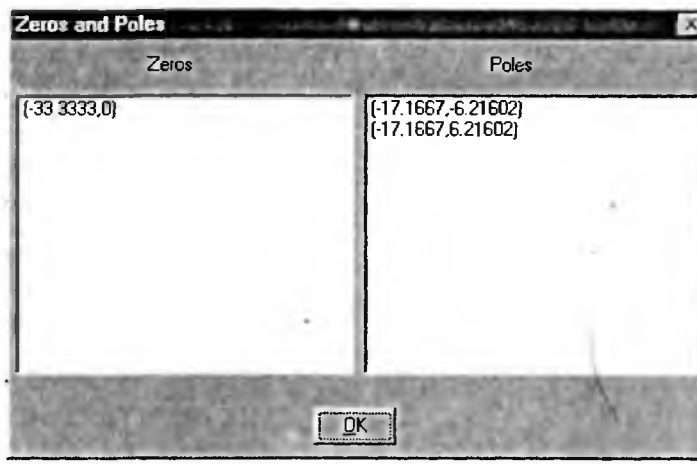


Рис. 5.17. Окно с данными о нулях и полюсах выделенной подсистемы

По завершении просмотра этих данных окно можно закрыть. При этом появится окно с графическим представлением нулей и полюсов. Их расположение имеет важное значение для оценки многих особенностей поведения систем. Их описание имеется в книгах по теории автоматического управления, но выходит за рамки данной книги, посвященной конкретным программным средствам.

5.2.5. Построение АЧХ, ФЧХ и диаграммы Найквиста

Столь же просто можно получить такие важные характеристики системы, как АЧХ, ФЧХ и диаграмма Найквиста (годограф АЧХ). Для этого надо исполнить команды Frequence Responce и Nyquist Responce. Окно анализируемой системы со всеми полученными в результате экспресс-анализа данными представлено на рис. 5.18.

Обращает на себя внимание высокое качество представления всех графических характеристик и их соответствие принятым в литературе и документации видам таких характеристик. Впрочем, есть возможность менять вид характеристик с помощью окон настройки. Одно из таких окон, вызываемое командой Root Locus Options, показано в верхней части экрана. Подобное окно для установки диапазона частот появляется при исполнении команды

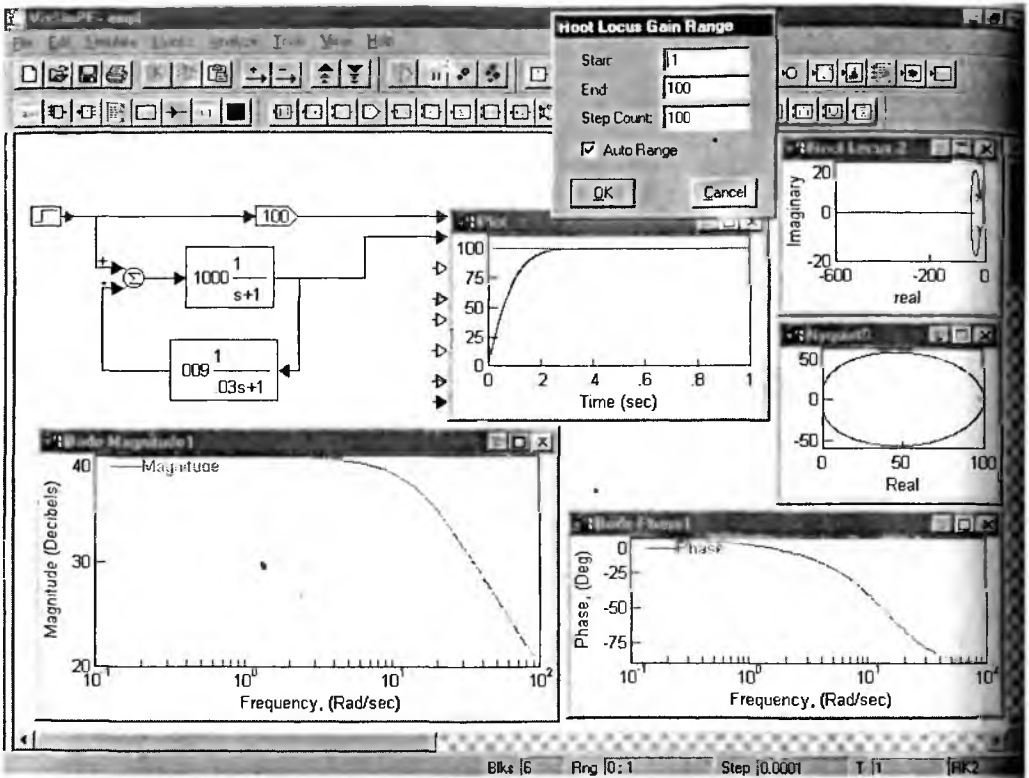


Рис. 5.18. Окно анализируемой системы с ее графическими характеристиками

Frequence Rate. В нем устанавливается начальная и конечная частоты и ша изменения частоты.

5.2.6. Применение графического маркера

Здесь уместно отметить, что построенные графики могут анализироватьс с помощью специального маркера, который можно вывести, активизиру кнопку Read Coordinates... в окне свойств графики. На рис. 5.19 эта возмож ность показана на примере вычисления частоты отсечки с помощью АЧХ сис

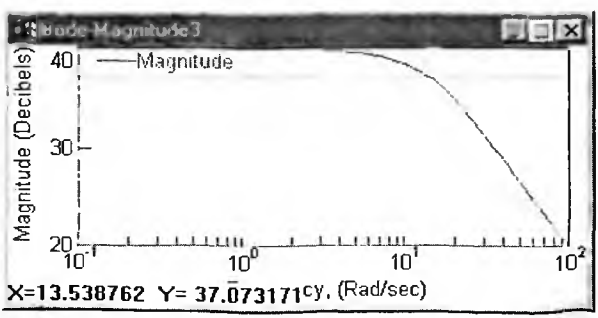


Рис. 5.19. Применение графического маркера

темы. Маркером в виде большого креста задается уровень около 37 дБ (на 3 дБ ниже уровня 40 дБ). Переместив маркер до кривой, можно считать частоту отсечки — она равна около 13,5 рад/с. Отсчет координат маркера дается под графиком.

5.2.7. Установка предпочтений Preferences

Для установки глобальных опций (предпочтений) средства Analyze служит команда Preferences. Она выводит окно, которое показано на рис. 5.20.

В этом окне возможны следующие установки:

- Digits of Display Precision — установка числа знаков отображения чисел;
- Frequence Unit — установка единицы измерения частоты (в рад/с или Гц);
- Phase — установка единиц измерения фазы (в радианах или градусах);
- Magnitude — изменение единицы измерения отношения сигналов на выходе и на входе (в Log 10 или в децибелах);
- Zero Theresold — установка порогов для числовых данных, после которых они принимаются за нулевые.

Эти установки вполне очевидны и в особых пояснениях не нуждаются. Исключением является последний параметр — его неправильная установка может вызвать чрезмерно большие ошибки в вычислении передаточных характеристик.

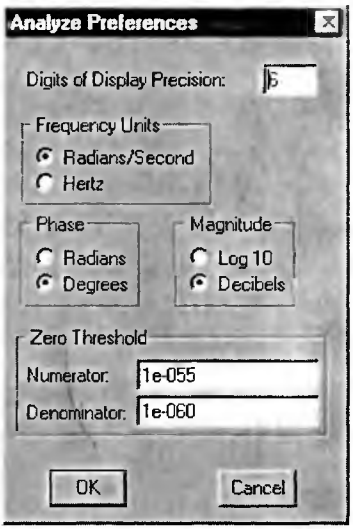


Рис. 5.20. Окно установки предпочтений средства Analyze

5.2.8. Конструирование компенсаторов

Средства Analyze позволяют также создавать *компенсаторы*, введение которых обеспечивает устойчивость изначально неустойчивых систем с обратной связью. К сожалению, этот процесс не автоматизирован и может выполняться только достаточно опытными пользователями. Поэтому рассмотрим конструирование компенсаторов с формальных позиций, полагая, что пользователь знает, что он делает.

Рисунок 5.21 показывает блок передаточной функции с подключенными к нему источником перепада и осциллографом. Передаточная характеристика блока соответствует неустойчивой системе — сигнал на выходе экспоненциально нарастает. Приведенные АЧХ и ФЧХ позволяют опытному пользователю оценивать запасы по устойчивости и фазе и выяснять требования к их компенсации.

Выделив систему и исполнив команду Compensator Design, можно получить вывод диалогового окна конструирования компенсаторов. Это окно показано на рис. 5.22 вместе с диаграммой модели и ее графическими характеристиками.

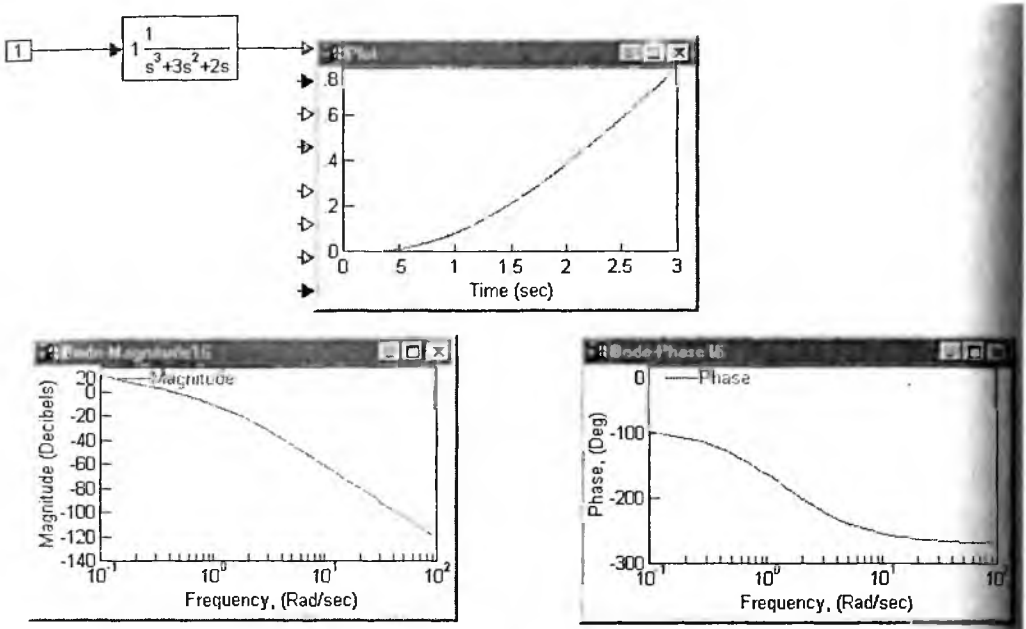


Рис. 5.21. Исходная система и его характеристики

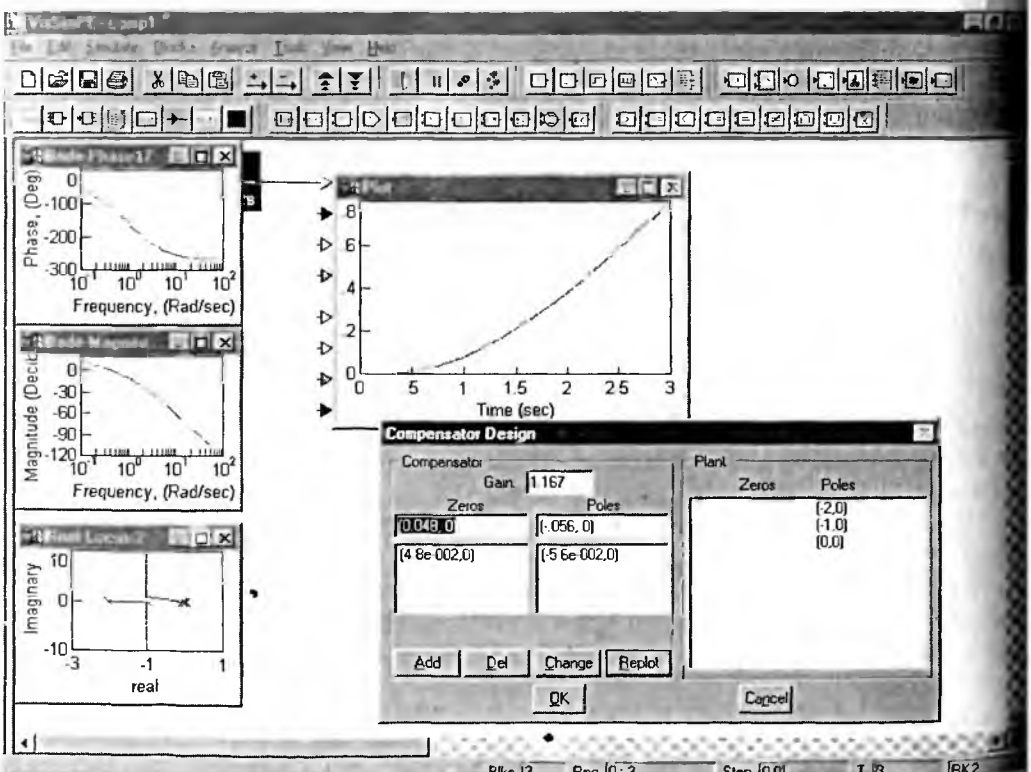


Рис. 5.22. Рабочий стол и окно конструирования компенсаторов

Задача конструктора компенсатора заключается в подборе методом проб и ошибок нулей и полюсов компенсатора. Для этого в окне компенсатора имеются соответствующие места ввода и кнопки для добавления данных в список нулей и полюсов (кнопка Add), стирания данных (кнопка Del) и их коррекции (кнопка Change). Кнопка Replot позволяет дополнить графики характеристик новыми кривыми. Наконец кнопка OK завершает этот процесс и приводит к появлению в окне модели блока компенсатора (рис. 5.23).

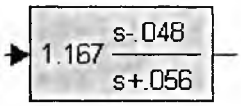


Рис. 5.23. Сконструированный блок компенсатора

5.3. Блоки для систем управления

Для создания систем управления могут использоваться блоки стандартной библиотеки блоков VisSim. Однако часто необходимы специальные блоки, которые можно найти в дополнительных библиотеках и примерах применения системы VisSim.

5.3.1. Блок пропорционального регулятора

Для создания систем с обратной связью приходится использовать структуру пропорционального регулятора, представленную на рис. 5.24. Она содержит суммирующе-вычитающее устройство с двумя входами и умножитель. Коэффициент передачи устройства задается параметром Proportional Gain, значение которого задано равным 1, но может меняться.

Организация линейной системы с помощью этого блока представлена на рис. 5.25. Здесь помимо блока Proportional Gain используется блок задания передаточной функции, что позволяет учесть инерционность системы.

Блок Proportional Gain ничего особого не делает. Он лишь упрощает построение линейных систем и делает их структурную схему более привычной для специалистов в области автоматического управления.

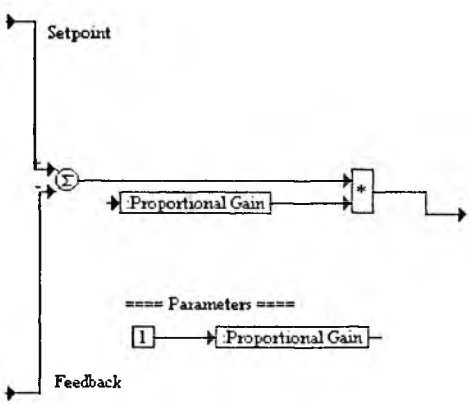


Рис. 5.24. Диаграмма блока пропорционального регулятора

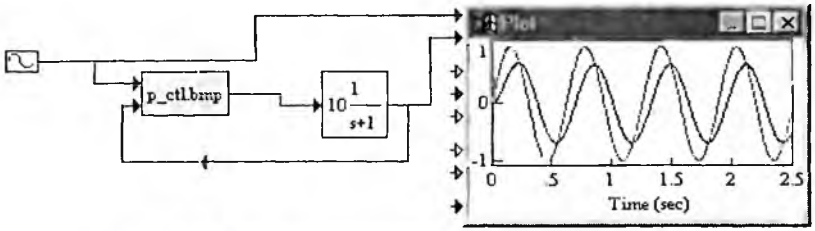


Рис. 5.25. Пример применения блока Proportional Gain

5.3.2. Блоки регуляторов ПИ и ПИД

Многие системы строятся с регуляторами, в которых, помимо пропорционального звена, включается интегрирующее звено. Такие регуляторы принято называть *пропорционально-интегрирующими* или регуляторами ПИ (PI) типа. Структурная схема блока PI Control такого типа представлена на рис. 5.26.

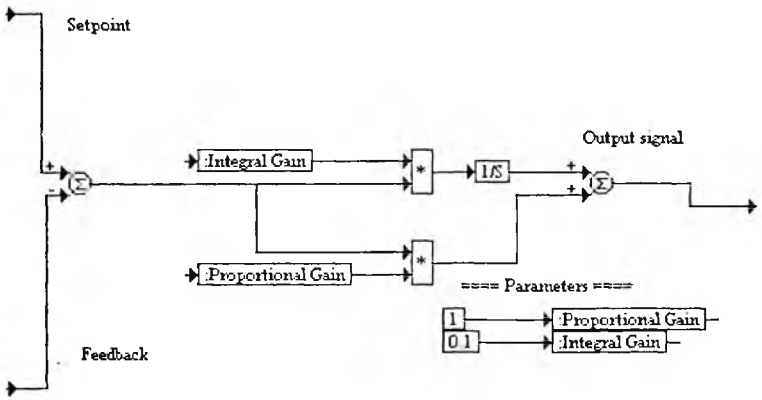


Рис. 5.26. Структурная схема блока PI Control

Рисунок 5.27 показывает работу такого блока при подаче на один вх прямоугольного импульсного сигнала, а на второй вход — константы – 3. Вы ходной сигнал обнаруживает как компоненту входного сигнала, так и тренд отрицательным наклоном, вызванный действием интегратора.

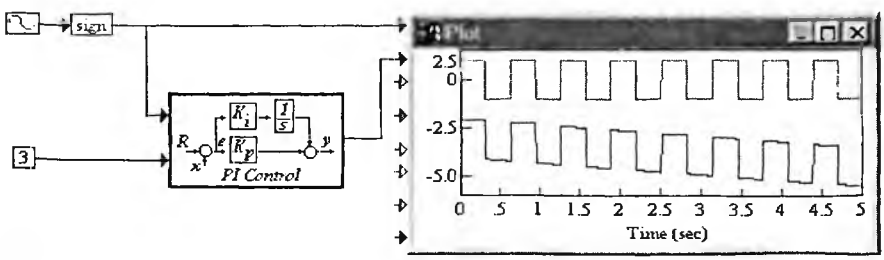


Рис. 5.27. Пример применения блока PI Control

В задачах управления объектами часто используются *пропорционально-интегро-дифференциальные регуляторы* ПИД (PID). Их передаточная характеристика записывается в виде:

$$y = (g_{\text{п}} + \frac{g_{\text{и}}}{s} + g_{\text{д}} \cdot s) \cdot x,$$

где $g_{\text{п}}$, $g_{\text{и}}$ и $g_{\text{д}}$ — коэффициенты передачи звеньев, соответственно пропорционального, интегрирующего и дифференцирующего звеньев PID-регулятора. Такие регуляторы реализует блок PID Control, представленый на рис. 5.28.

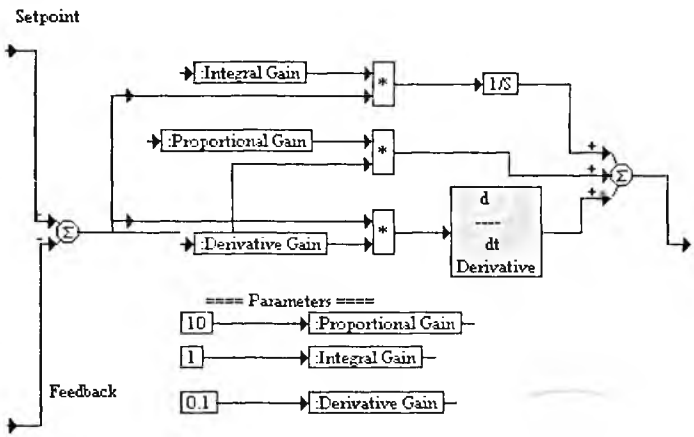


Рис. 5.28. Структурная схема блока PID Control

Применение блока PID Control заметно упрощает построение систем на базе ПИД-регуляторов. Пример применения блока PI Control представлен на рис. 5.29. Здесь система реализована на базе ПИД-регулятора и блока передаточной функции первого порядка. На вход подается сигнал, полученный вычислением абсолютного значения синусоидального сигнала. Он имитирует сигнал с частотой 20 рад/с на выходе двухполупериодного выпрямителя без фильтра. Любопытно отметить, что данная система дает на выходе почти синусоидальный сигнал, несмотря на то что форма входного сигнала далека от синусоидальной.

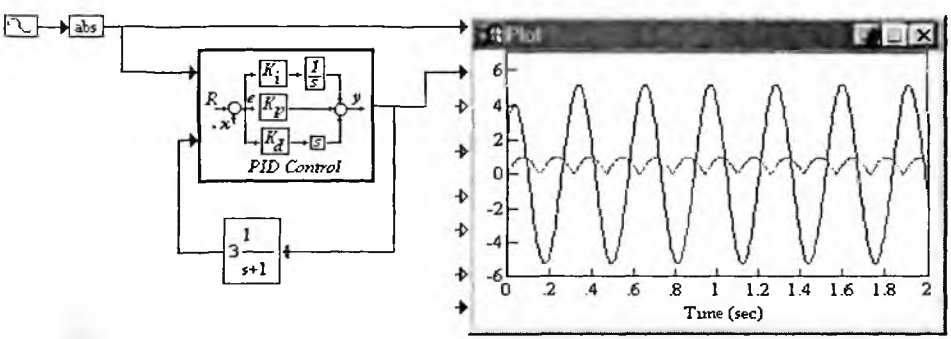


Рис. 5.29. Пример применения блока PID Control

5.3.3. Блок Rate Feedback Control

Блок Rate Feedback Control представляет собой регулятор, блок-схема которого представлена на рис. 5.30. Этот блок находит ограниченное применение в тех системах, где его структура приемлема и целесообразна.

Пример применения блока Rate Feedback Control представлен на рис. 5.31. Нетрудно заметить, что реакция блока на прямоугольные импульсы складывается из самых импульсов и компоненты дифференцирования, обусловленной инерционностью внешнего блока обратной связи.

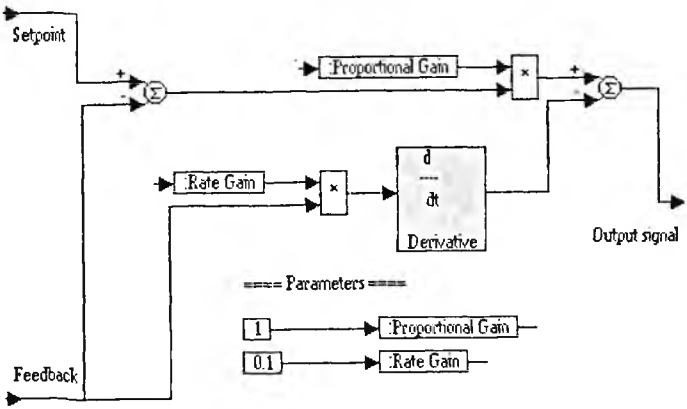


Рис. 5.30. Структурная схема блока Rate Feedback Control

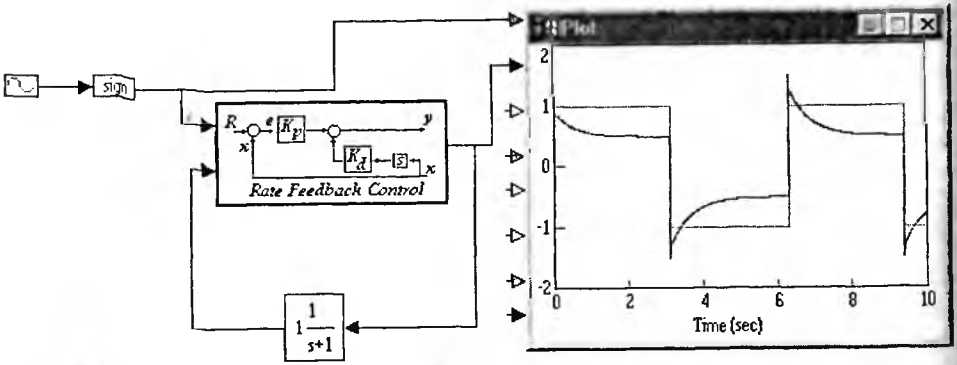


Рис. 5.31. Пример применения блока Rate Feedback Control

5.4. Средства оптимизации параметров систем

5.4.1. Простой пример на оптимизацию

Основные блоки оптимизации мы уже рассмотрели в главе 3. Продолжим эту важную тему и рассмотрим демонстрационные примеры на ее применение. Один из простейших примеров представлен на рис. 5.32. Это пример на минимизацию параболической поверхности.

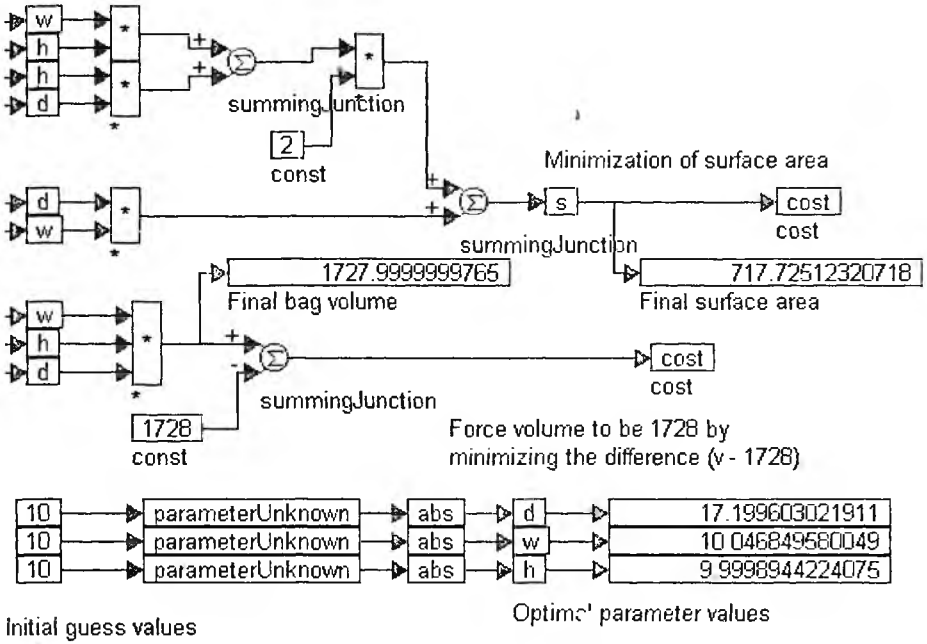


Рис. 5.32. Пример на минимизацию поверхности

Минимизация задается блоками cost, а для задания исходных значений параметров служат блоки parametrUnknown. Индикация текущих и окончательных значений параметров реализована виртуальными цифровыми измерителями уровня сигналов. Этот пример настолько прост, что его стоит внимательно изучить самостоятельно.

5.4.2. Аппроксимация полуволны синуса двумя отрезками прямой

При реализации систем контроля часто возникает необходимость в приближении некоторых зависимостей. Если система реализуется средствами микроэлектроники, то надо предельно упрощать реализацию приближения. Достаточно часто используется аппроксимация функции синуса, точнее, одной ее полуволны (вторая получается сдвигом и симметричным отражением первой полуволны).

Рисунок 5.33 служит для оптимизации приближения полуволны синуса двумя отрезками прямых. Требуется так подобрать положение двух отрезков прямой, чтобы среднеквадратическая ошибка приближения синуса оказалась минимальной.

В этом примере для каждого отрезка предусмотрено изменение наклона (параметры a и b) и сдвига во времени (задержки delay). При пуске данной модели можно наглядно наблюдать перемещение обоих отрезков до достижения минимума среднеквадратической ошибки. Блок оптимизации задан в левом нижнем углу рис. 5.33.

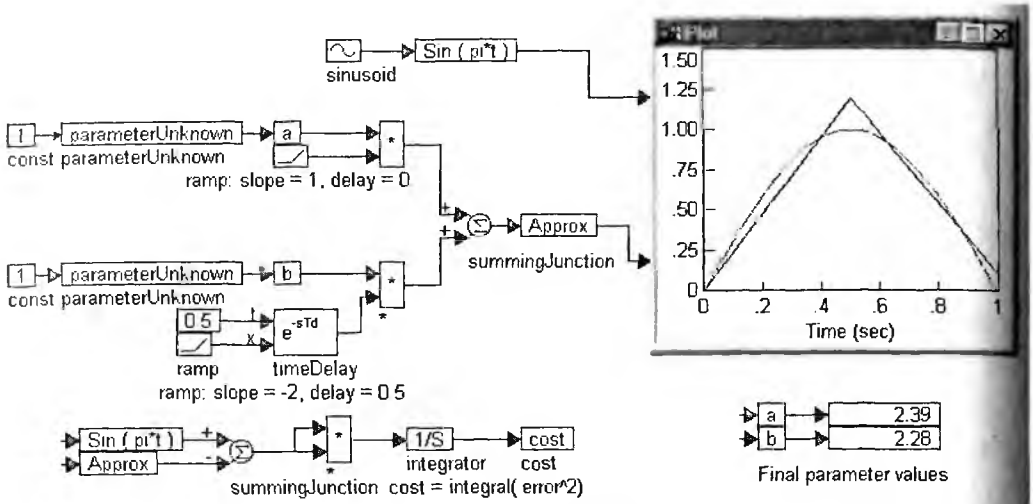


Рис. 5.33. Пример на оптимизацию приближения полуволны синуса двумя отрезками прямых

5.4.3. Аппроксимация полуволны синуса четырьмя отрезками прямой

Теперь усложним нашу задачу и рассмотрим оптимизацию приближения полуволны синуса четырьмя отрезками прямых. Необходимая для этого модель представлена на рис. 5.34.

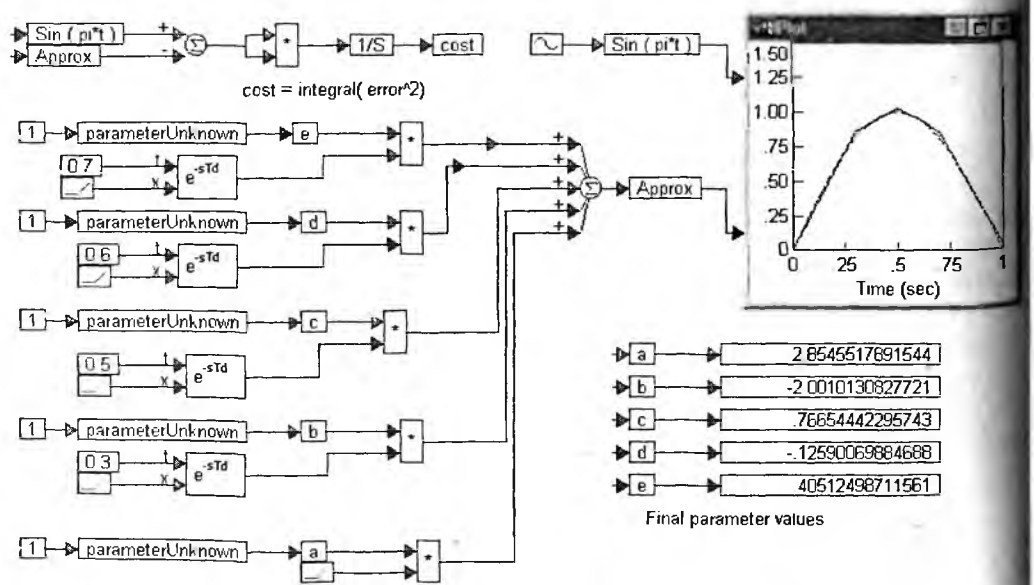


Рис. 5.34. Пример на оптимизацию приближения полуволны синуса четырьмя отрезками прямых

Нетрудно заметить, что модель рис. 5.34 построена по тому же принципу, что и в предшествующем примере. Просто возросло число блоков задания отрезков. Блок оптимизации переместился в верхний левый угол.

5.4.4. Оптимизация системы с ПИД-регулятором

Теперь рассмотрим более сложный пример. Рисунок 5.35 показывает типичную модель системы с ПИД-регулятором. Модель представляет собой типичную линейную замкнутую систему, имеющую заданную блоком Plant передаточную характеристику и ПИД-регулятор в цепи обратной связи. Назначение модели — создание системы, переходная характеристика которой должна быть монотонной и обеспечивать минимальное время нарастания.

Решается задача оптимизации коэффициентов g_p, g_u и g_d, которые в этой модели обозначены как P, I и D. Критерием оптимизации является получение минимального времени установления переходного процесса при заданном выбросе на вершине в 10%. Процесс начинается при значениях P, I и D, равных 1, а заканчивается при оптимальных значениях этих коэффициентов. Кривые погрешности и переходных процессов показаны также на рис. 5.35.

Построение субмодели ПИД-регулятора показано на рис. 5.36. Этот блок можно (выделив мышью и поместив в буфер командой Copy) использовать и для построения моделей других систем с ПИД-регулятором.

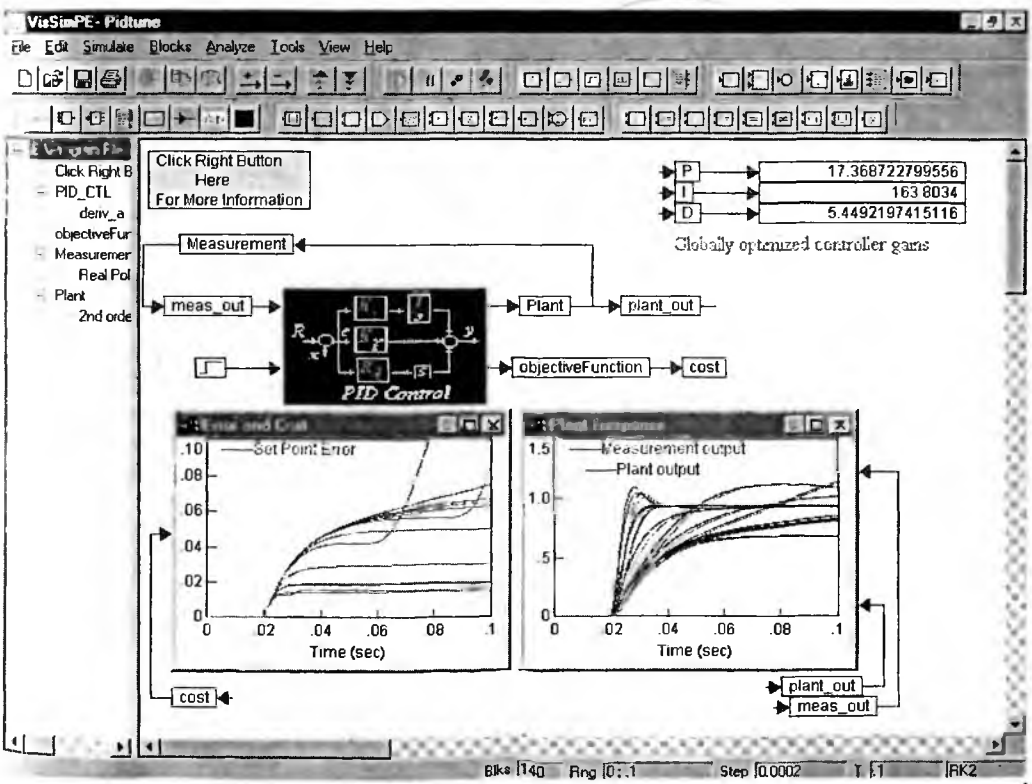


Рис. 5.35. Модель системы с ПИД-регулятором

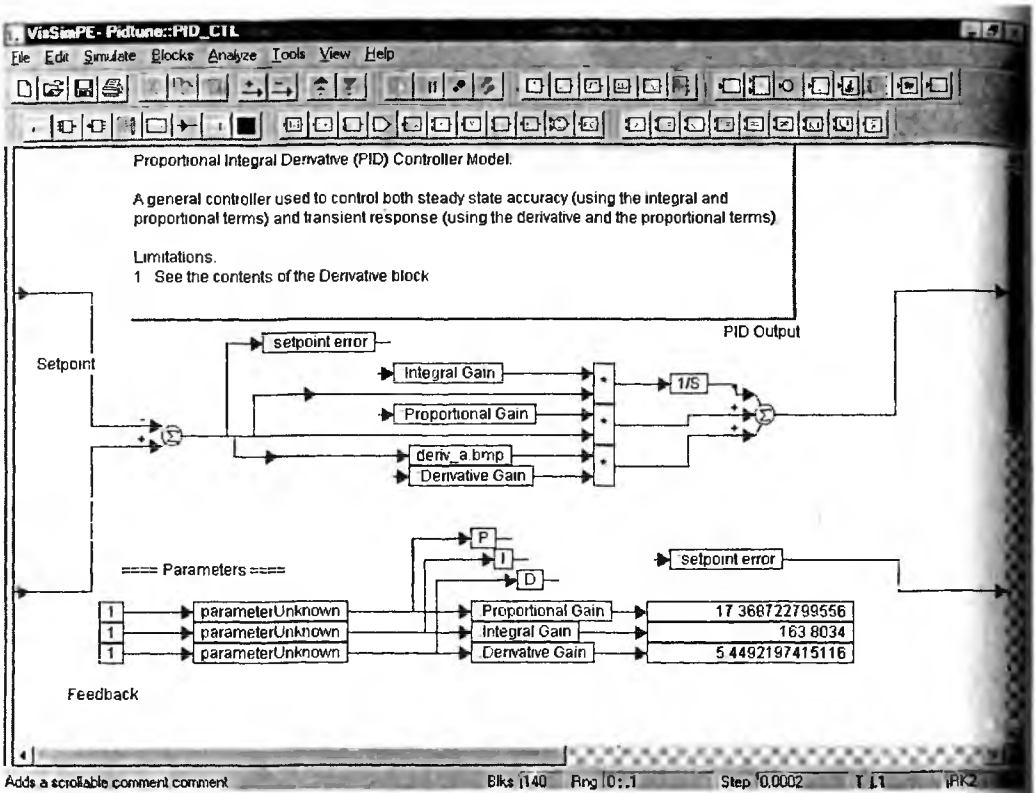


Рис. 5.36. Модель ПИД-регулятора

Знакомство с некоторыми деталями этой модели представляет большой интерес. Так, на рис. 5.37 представлена модель дифференцирующего устройства, которая в библиотеке блоков системы VisSim отсутствует. Из представленной модели следует, что дифференцирующее устройство можно создать из усиливающего сигнал тракта путем размещения в петле отрицательной обратной связи идеального интегратора. Модель последнего есть в библиотеке блоков VisSim.

Рисунок 5.38 показывает субмодель objectiveFunction, в которой задана целевая функция и блок cont для оптимизации. Благодаря средствам оптимизации модель рис. 5.35 обеспечивает неоднократное вычисление переходной характеристики данной замкнутой системы и выводит графики ошибки и графики вычисленных и измеренных переходных характеристик. Они показаны на рис. 4.15 на момент окончания оптимизации. Диаграмма субмодели блока сравнительных измерений Meassurement показана на рис. 5.39.

Из других блоков заслуживает внимание реализация блока Plate, в котором задается вполне определенная передаточная характеристика. Реализованное в блоке задание передаточной характеристики является альтернативным вариантом применения блока transferFunction. Диаграмма субмодели Plate представлена на рис. 5.40.

С оставшимися простыми блоками читателю несложно разобраться самостоятельно.

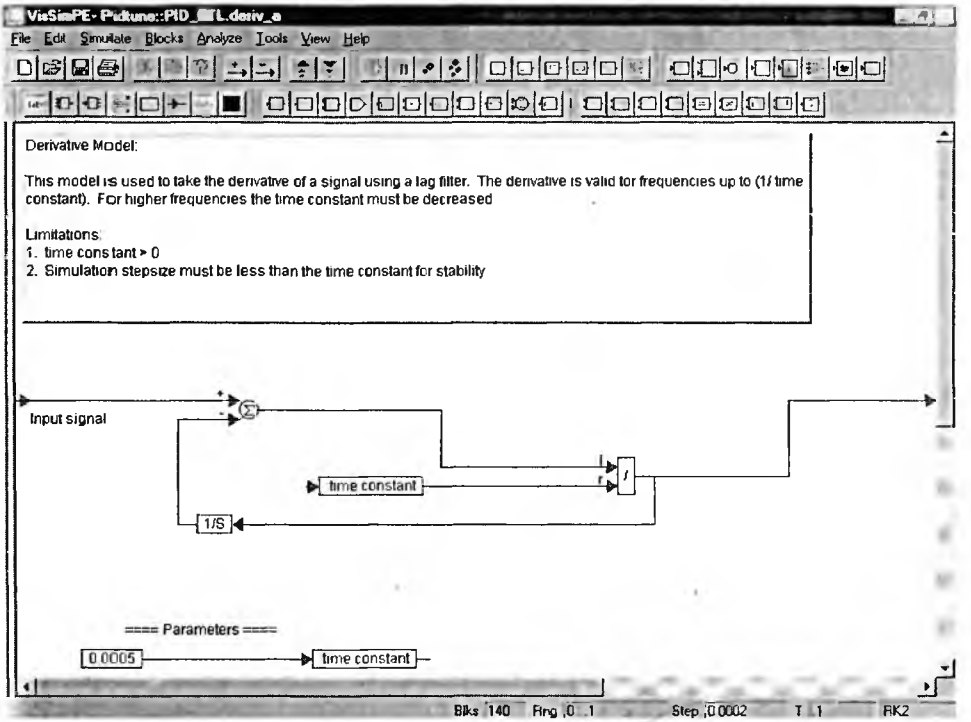


Рис. 5.37. Модель дифференцирующего устройства

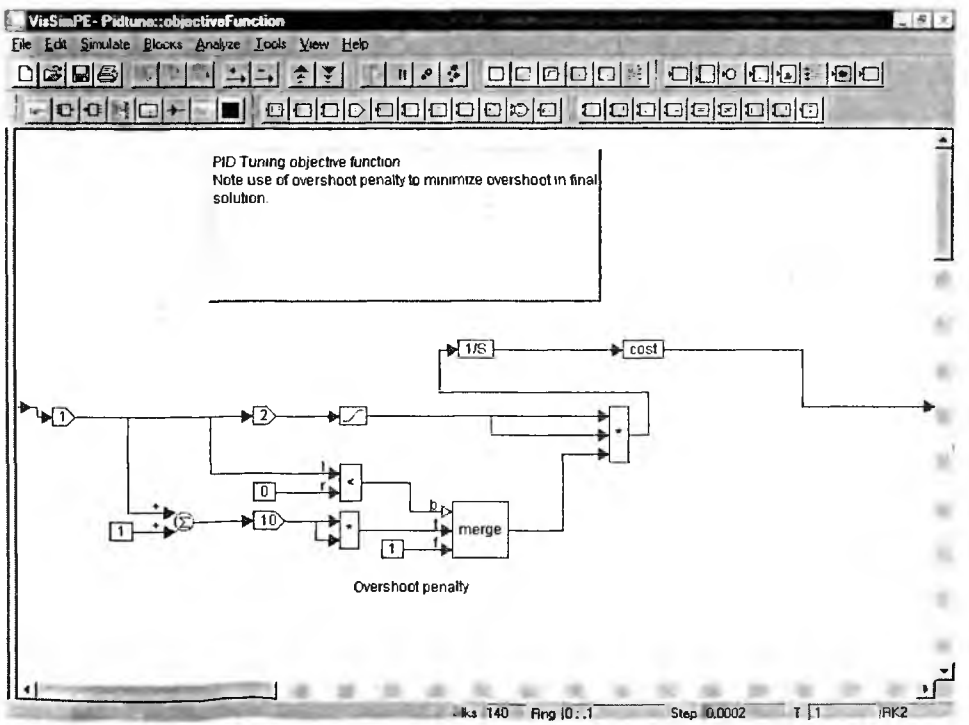


Рис. 5.38. Субмодель objectiveFunction

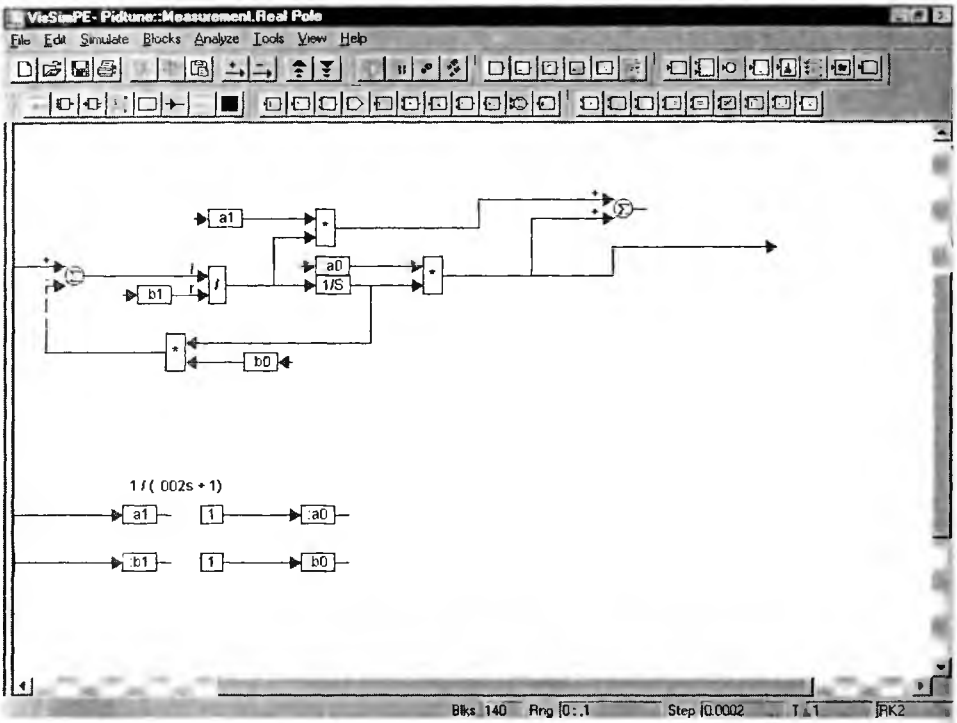


Рис. 5.39. Диаграмма блока измерений Meassurement

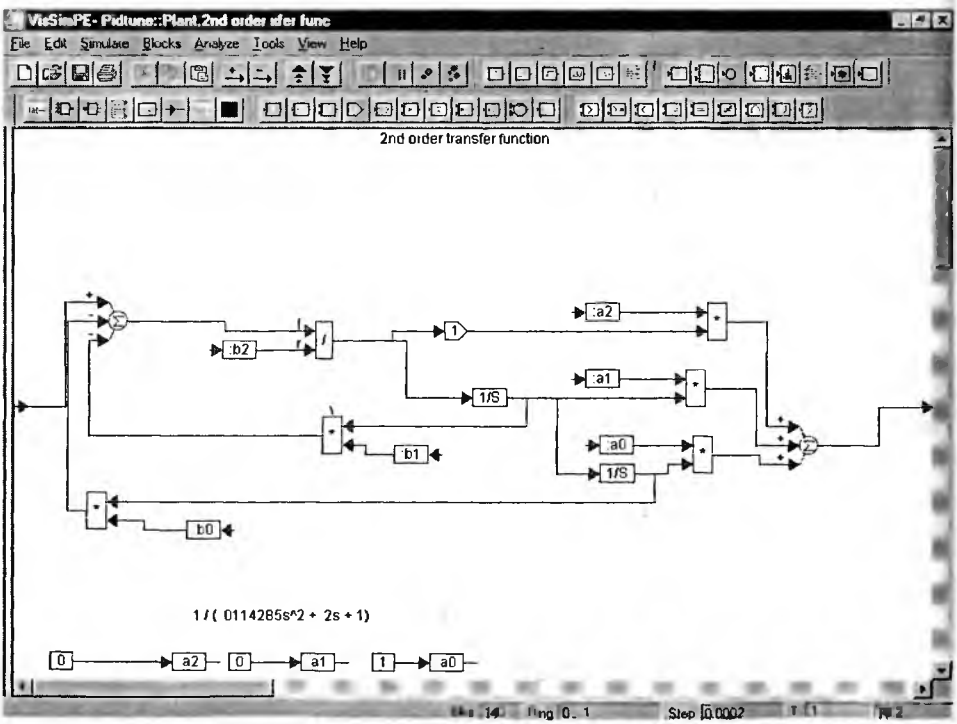


Рис. 5.40. Диаграмма субмодели Plate

5.5. Моделирование нелинейных систем второго порядка

В этом разделе мы рассмотрим несколько типовых задач на моделирование нелинейных систем второго порядка. Такие системы находят исключительно широкое применение. Достаточно отметить, что на них основаны многочисленные варианты колебательных систем и устройств начиная от механического маятника часов, LC-генераторов синусоидальных колебаний и высокоточных часов на основе кварцевых, квантово-механических и оптических резонаторов. Так что их моделирование может принести большую познавательную пользу.

5.5.1. Движение брошенного на пол мяча

К широко распространенным задачам на поведение систем второго порядка является задача о движении брошенного с высоты h на пол мяча. Поведение мяча общеизвестно — он, ударяясь о пол, подпрыгивает, но каждый раз на все более низкую высоту. Причиной затухания колебаний является прежде всего вязкость воздуха. Модель поведения мяча, имеющаяся в файле Bounce.vsm, представлена на рис. 5.41.

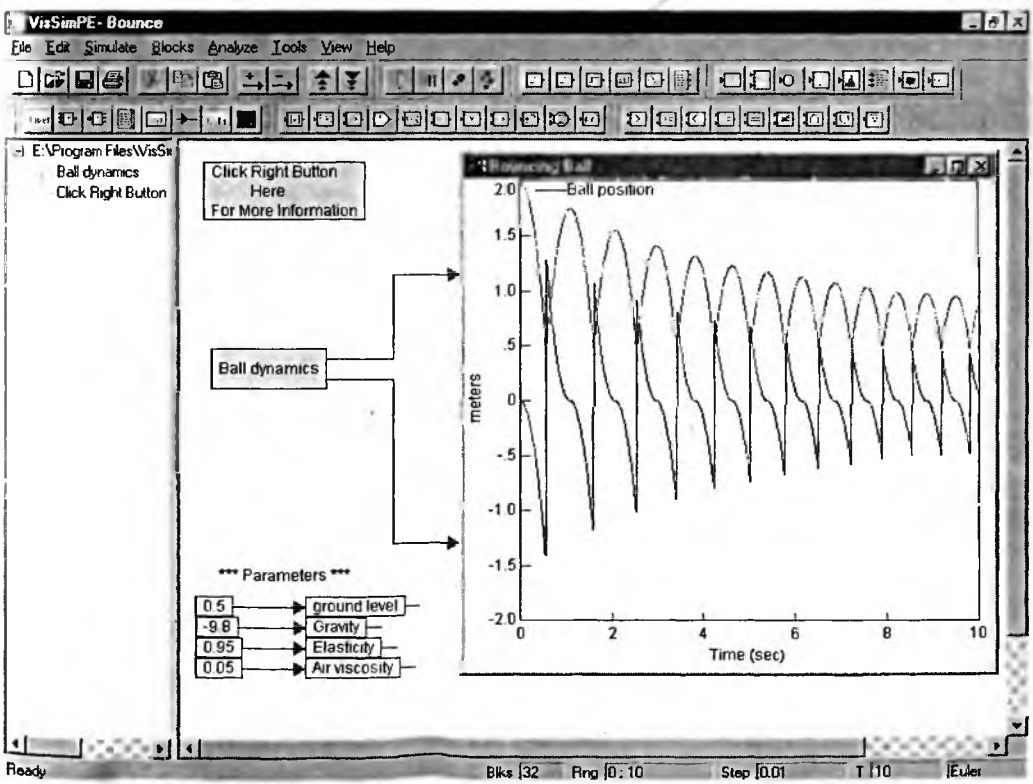


Рис. 5.41. Общая модель движения брошенного на пол мяча

Общая модель рис. 5.41 содержит единственный содержательный блок — детальную модель, которая показана на рис. 5.42. Кроме того, в общей модели заданы 4 глобальные переменные: уровень земли (пола) ground level, ускорение свободного падения (гравитационная постоянная) Gravity, эластичность Elasticity и вязкость воздуха Air viscosity. При активизации единственного блока модели Ball Dynamics (Динамика мяча) можно получить доступ к детальной модели динамики мяча, показанной на рис. 5.42.

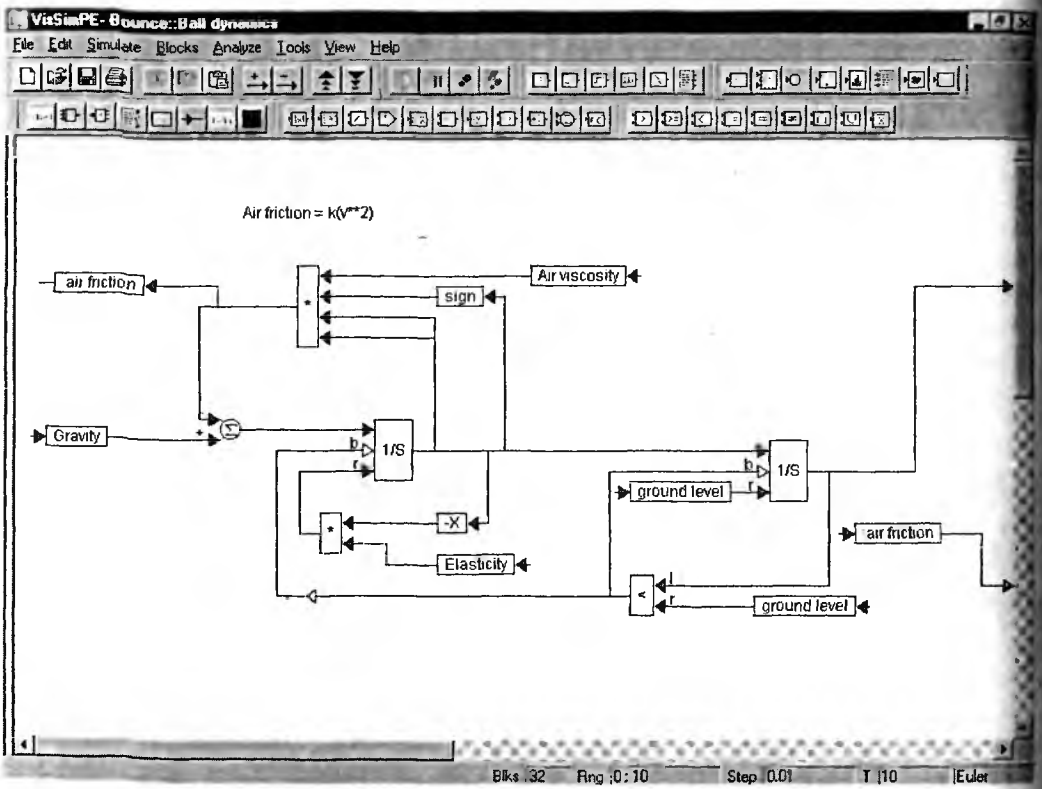


Рис. 5.42. Диаграмма модели Ball Dynamics

Модель динамики мяча построена на двух интеграторах. Недостающий важный параметр — высота h задается как Initial Condition = 2 для второго интегратора (он справа). Для первого интегратора Initial Condition = 0. Меняя исходные данные и значение h, можно получить множество временных зависимостей позиции мяча Ball Position и сопротивления воздуха Air Friction.

5.5.2. Моделирование движения мяча, подброшенного вверх

Теперь решим еще одну схожую задачу — моделирование движения мяча, который подброшен вверх с высоты $h = 10$ м со скоростью 15 м/с. Если убрать «излишества» предшествующей модели, то модель данной задачи можно свести к единственной блок-схеме, представленной на рис. 5.43. Здесь также высота h задается как параметр Initial Condition = 10 для второго интегратора.

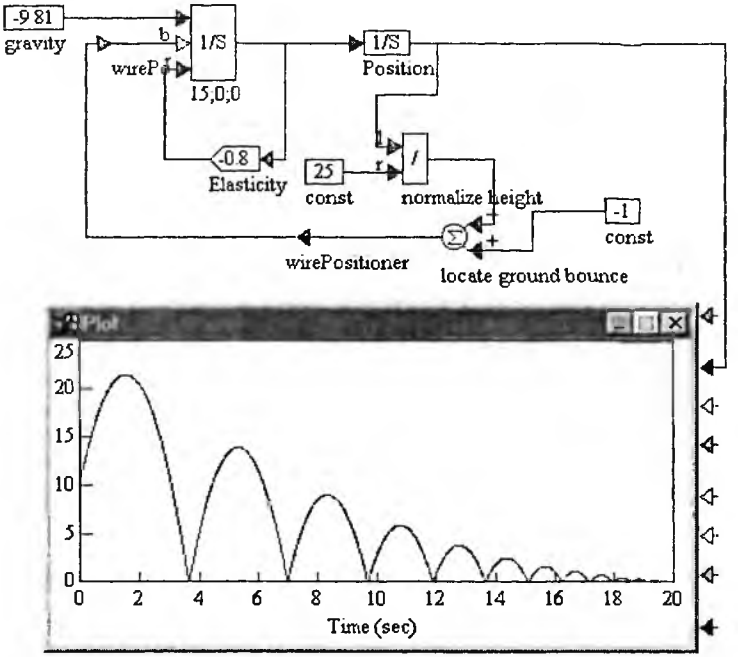


Рис. 5.43. Модель движения мяча, подброшенного вверх

В этом примере особенно отчетливо видно, что частота подпрыгивания мяча растет во времени. Хотя задача несколько усложнена по сравнению с ее первым вариантом, реализация модели сделана несколько проще. Это также способствовало возможности размещения ее в одном окне VisSim.

5.5.3. Колебания груза на пружине с демпфированием

Демпферы железнодорожных вагонов и автомобилей часто используют пружины для демпфирования колебаний маятников, которые образуются массой, прикрепленной к пружине. Для моделирования подобных систем используются модели маятников. Одна из таких моделей представлена на рис. 5.44 вместе с иллюстрацией, дающей наглядное представление о механическом маятнике.

Поведение маятника определяется тремя константами: массой колеблющегося объекта Mass, Spring constant и постоянной затухания Damping constant. Все они задаются ползунковыми регуляторами, показанными на рис. 5.44. Осциллограф позволяет наблюдать временные зависимости для положения центра массы, скорости и ускорения. В данном случае мы имеем типичные затухающие колебания.

Щелкнув мышью по рисунку, можно вывести блок-схему данной модели (рис. 5.45), которая содержит блоки, решающие нелинейное дифференциальное уравнение маятника. Оно представлено в текстовой вставке. Осциллограф на рис. 5.45 строит более детальную осциллограмму для зависимости положения центра массы от времени.

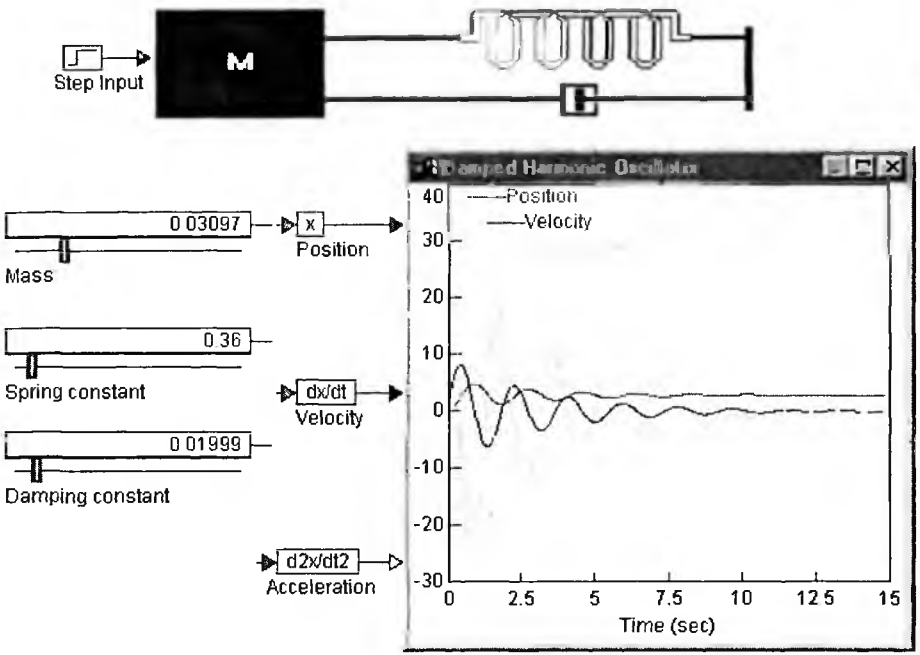


Рис. 5.44. Модель механического маятника типа «масса на пружине»

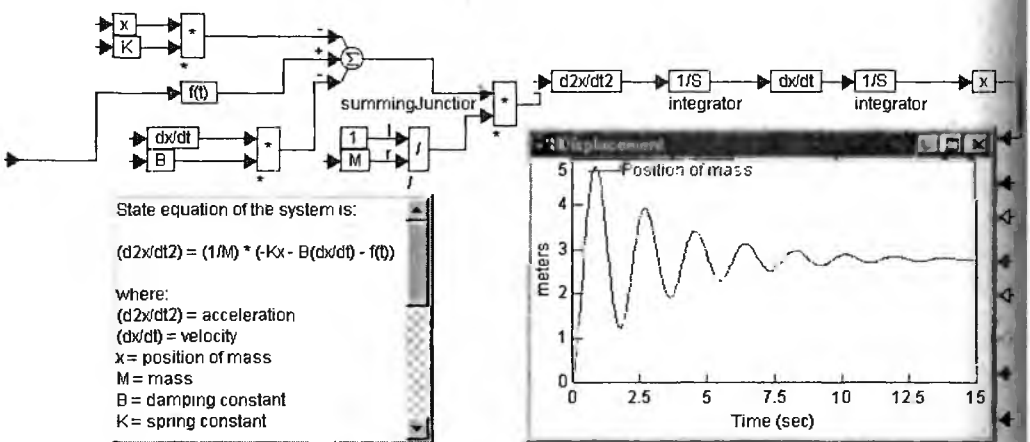


Рис. 5.45. Блок-схема модели демпфированного маятника

5.5.4. Моделирование колебательной системы Ван дер Поля

Другим классическим примером реализации нелинейной системы второго порядка является система, которая описывается следующим нелинейным дифференциальным уравнением Ван дер Поля:

$$\frac{d^2x(t)}{dt^2} - mu \cdot (1 - x(t)^2) \cdot \frac{dx(t)}{dt} + x(t) = 0.$$

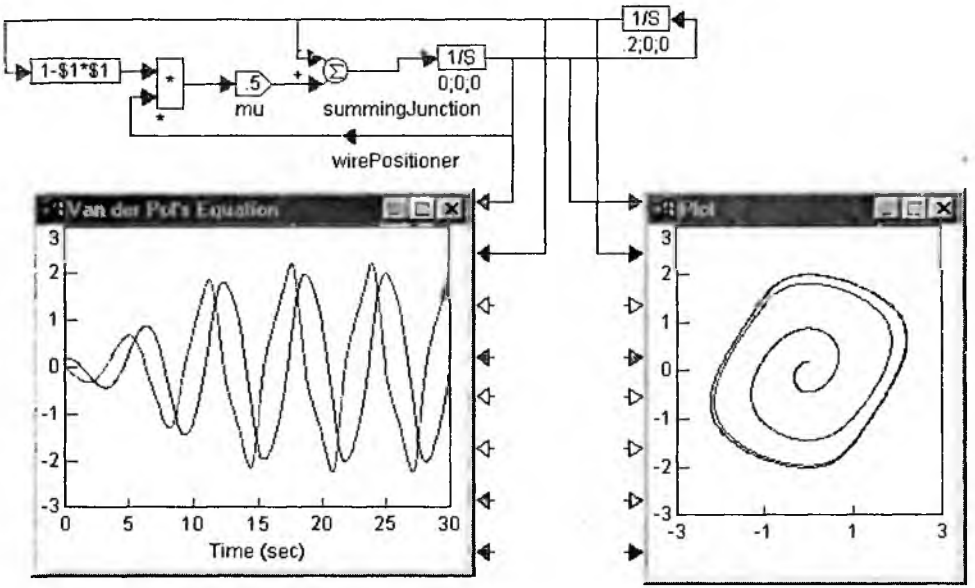


Рис. 5.46. Модель колебательной системы Ван дер Поля

Равенство нулю правой части этого уравнения свидетельствует о том, что система не находится под влиянием внешнего воздействия и является автономной. Тем не менее в такой системе могут возникать как затухающие, так и нарастающие и переходящие в стационарные колебания. Система Ван дер Поля неплохо описывает колебания в электронных генераторах на электронных лампах и транзисторах, что и объясняет ее широкую известность.

Реализовать решение уравнения Ван дер Поля несложно с помощью двух дифференцирующих объектов на базе интеграторов и блоков умножения и сложения. Достаточно наглядная и простая модель системы Ван дер Поля представлена на рис. 5.46. Существенное упрощение этой модели (по сравнению с имеющейся в демонстрационных примерах) достигнуто минимизацией блоков и реализацией нелинейной зависимости $(1- x(t)^2)$ блоком выражения, а не набором арифметических блоков.

Характер возникающих в системе Ван дер Поля колебаний зависит от начальных значений x и производной, а также от параметра *mu*. Начальные значения x и x' задаются параметром Initial Condition интеграторов, а параметр *mu* коэффициентом передачи блока Gain. На рис. 5.46 под моделью представлены временные зависимости $x(t)$ и $x'(t)$, а также фазовый портрет колебаний — он строится при подаче этих сигналов на входы горизонтального и вертикального отклонения осциллографа (опция XY Plot в окне свойств осциллографа).

Нетрудно заметить, что при представленных исходных данных колебания в системе нарастают по амплитуде и быстро приближаются к стационарным. Из-за отмеченной выше нелинейности форма колебания заметно отличается от синусоидальной, хотя и не имеет еще разрывного характера, наблюдающегося у релаксационных устройств.

5.5.5. Аттрактор Лоренца

Чем сложнее система и чем большим количеством дифференциальных уравнений она описывается, тем больше вероятность возникновения в системе хаотических режимов, даже если она автономна. Изучение этого вопроса показало, что уже в системах из трех дифференциальных уравнений возможно возникновение хаотических режимов. Наглядным примером этого является аттрактор Лоренца, поведение которого зависит от параметров *a*, *b* и *c*. При определенных значениях этих параметров и начальных параметров переменных поведение аттрактора (он в этом случае называется *странным аттрактором*) очень напоминает хаотические колебания [38].

Аттрактором в теории колебаний называется притягивающая область в фазовом пространстве. Причины неустойчивости аттракторов связаны с экспоненциальной неустойчивостью системы в малых областях фазового пространства. При этом наблюдаются хаотические переходы из одной области фазового пространства в другие, но при этом колебания могут не выходить из некоторой более обширной области фазового пространства. «Обвал» системы означает переход в некоторое состояние, резко отличающееся от других состояний, т. е. выход за пределы ограниченного фазового состояния системы. Такое состояние может оказаться устойчивым и привести к переходу системы в статическое состояние, при котором изменения ее параметров отсутствуют.

Укрупненная модель аттрактора Лоренца представлена на рис. 5.47. Модель содержит блок Lorenz, задающий уравнения аттрактора как систему из трех дифференциальных уравнений первого порядка, блок задания исходных параметров *a*, *b* и *c* и блок осциллографа для построения фазовых портретов колебаний в виде проекций пространственного фазового портрета.

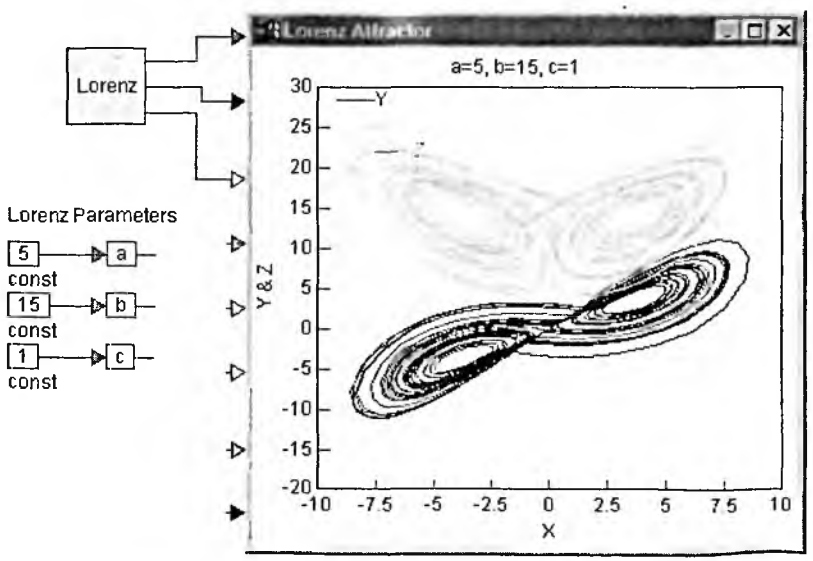


Рис. 5.47. Укрупненная модель аттрактора Лоренца

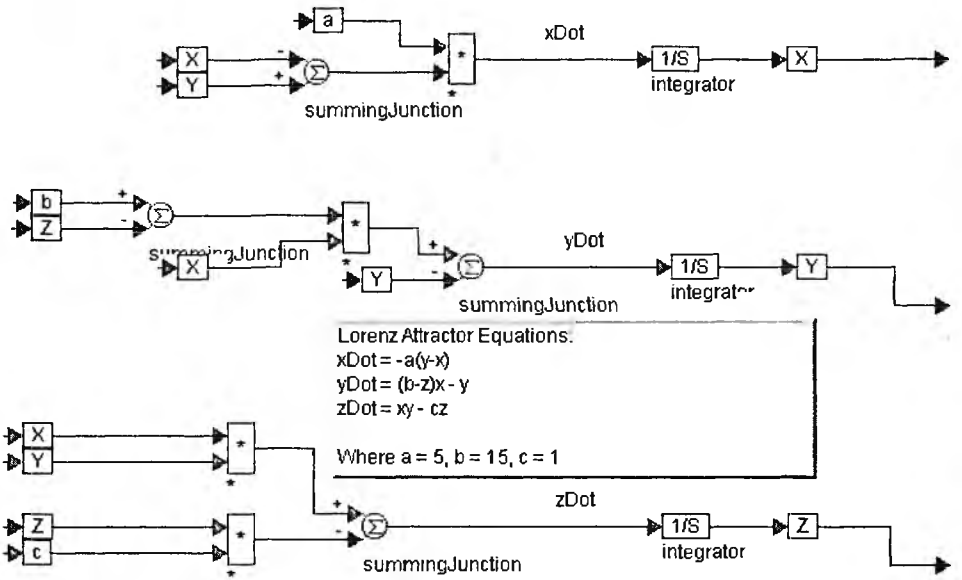


Рис. 5.48. Диаграмма модели аттрактора Лоренца

При активизации блока Lorenz появляется диаграмма с детальной моделью аттрактора Лоренца. Она показана на рис. 5.48. Помимо блоков, реализующих решение системы уравнений Лоренца и дающих вывод временных зависимостей *x*, *y* и *z*, в диаграмме присутствует текстовый блок с кратким описанием аттрактора Лоренца и уравнений этого аттрактора.

5.6. Демонстрационные примеры VisSim

В папке Appexmpl находится ряд папок с демонстрационными примерами системы VisSim. Эти примеры рассчитаны на самостоятельное знакомство с ними пользователя системы. Разумеется, знакомство рационально с избранными примерами, а не со всеми подряд. Дабы не лишать читателя возможности самостоятельно просмотреть представленные примеры, кратко познакомимся лишь с наиболее важными из них.

5.6.1. Моделирование полета летательного объекта

Моделирование полета летательного объекта — один из самых сложных из набора примеров, поставляемых с VisSim. Он расположен в папке Aerospac и представлен файлом 6dof.vsm. Документация по этому примеру с подробным изложением основ 3D-геометрии и динамики движения летательных аппаратов имеется в папке с документацией, которая создается при инсталляции VisSim 4.5.

Рисунок 5.49 показывает основную диаграмму примера. В левой части экрана видно дерево объектов модели данного примера. Эта модель содержит

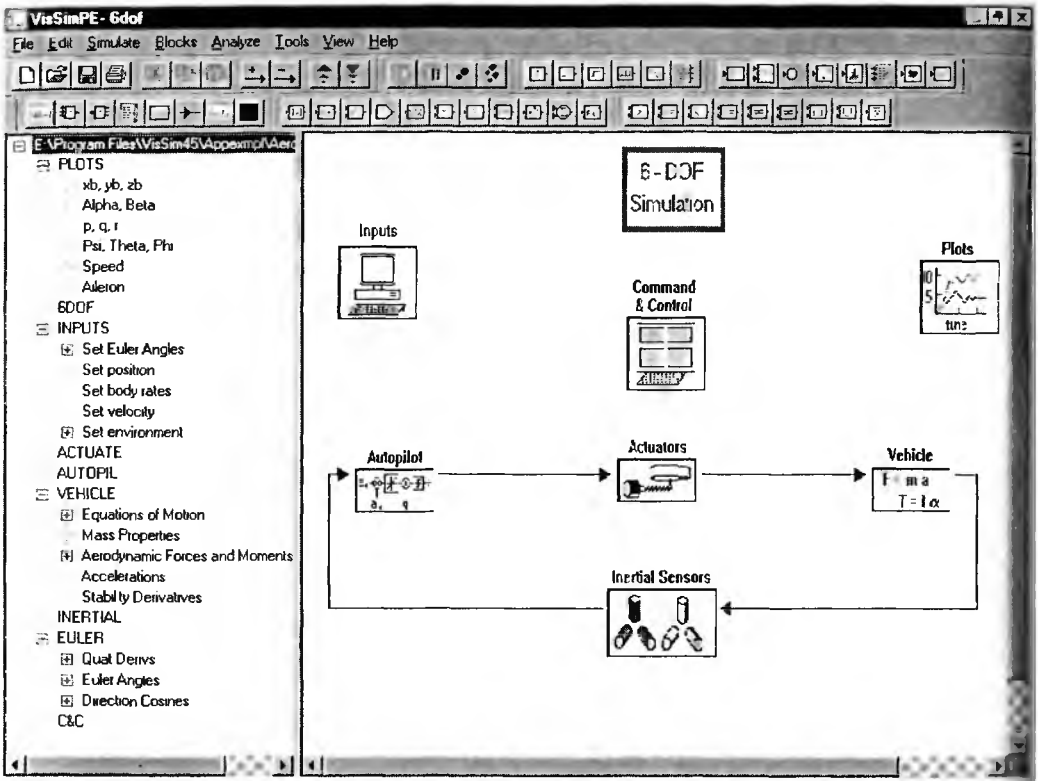


Рис. 5.49. Основная диаграмма примера 6-DOF Simulation

свыше полусотни подмоделей и является наглядным примером имитационного моделирования многоблочной сложной системы. Правда, объективности ради, стоит отметить, что некоторые субблоки очень просты и содержат всего 2—3 блока и соединения.

Блок Input дает доступ к блокам ввода исходных данных, блок Command — к средствам управления полетам и блок Plots — к графической иллюстрации моделирования. Под этими блоками расположена укрупненная блок-схема управления летательным аппаратом с помощью автопилота. Активизируя любой блок мышью, можно получить доступ к графической диаграмме подмодели, заданной соответствующим блоком.

В качестве примера доступа к подмоделям на рис. 5.50 представлена диаграмма подмодели Quat Derivs, в свою очередь содержащая 6 подмоделей. Данная подмодель не имеет входов, но имеет 4 вывода.

Графическая визуализация моделирования представлена разделом основной диаграммы Plots и базируется на применении виртуальных осциллографов (рис. 5.51). Верхняя осциллограмма характеризует динамику изменения одного из углов Phi (два другие Psi и Theta), а нижняя — вертикальную — проекцию профиля траектории.

Более подробное знакомство с этим примером вы можете провести самостоятельно, если эта область моделирования представляет для вас интерес.

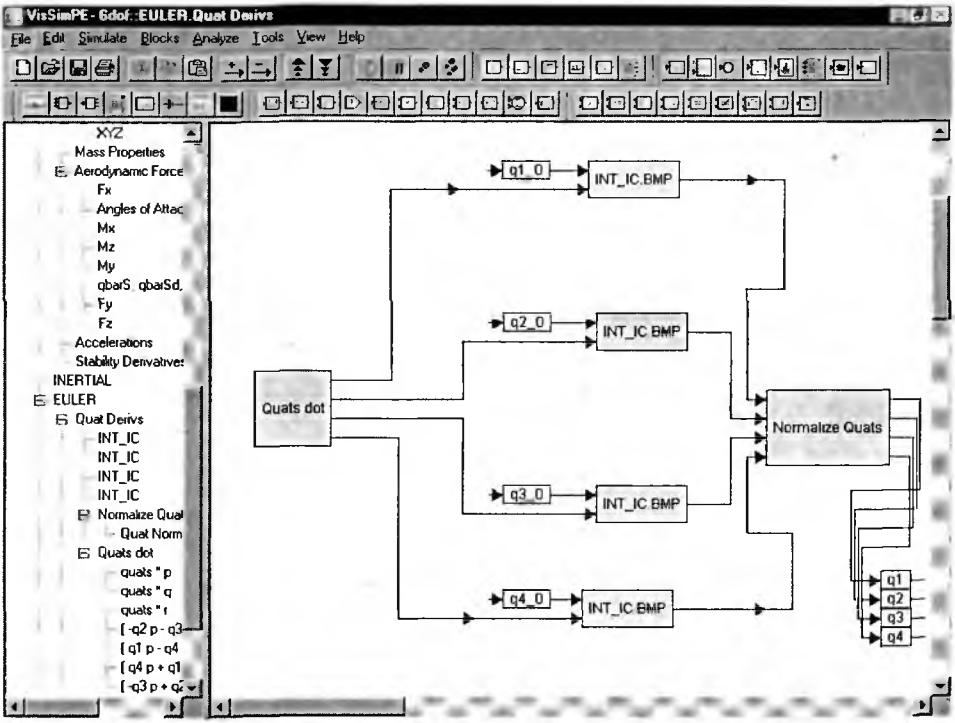


Рис. 5.50. Пример вывода подмодели Quat Derivs, имеющей только выходы

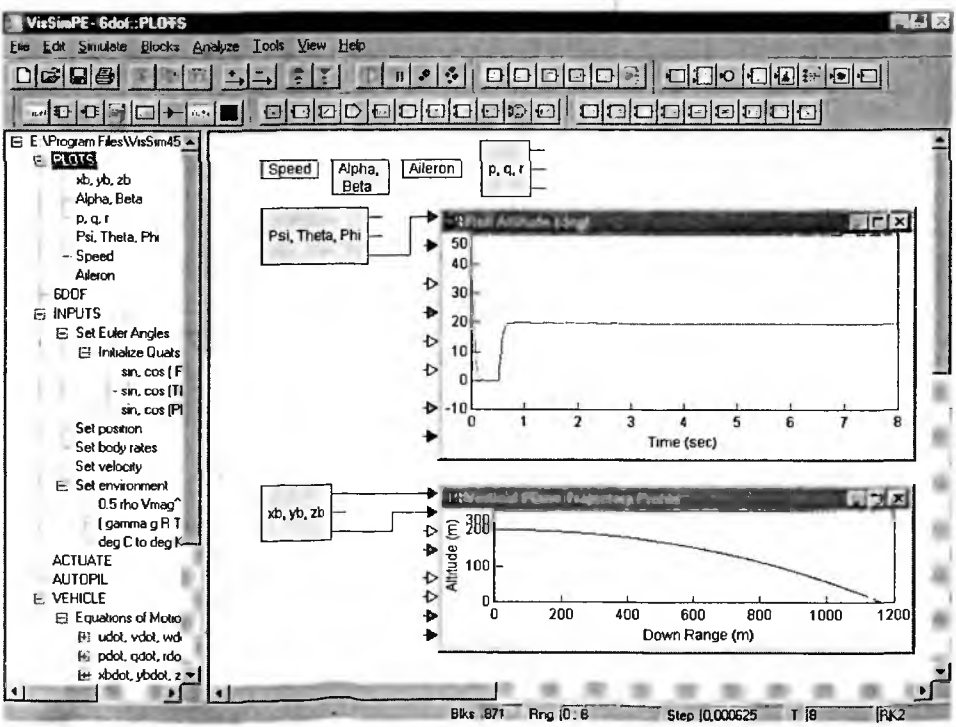


Рис. 5.51. Средства графической визуализации моделирования в модели 6-DOF Simulation

5.6.2. Пример на анимацию маятника из двух стержней

Средства анимации в VisSim весьма просты, но позволяют решать многие, в том числе учебные, задачи. В главе 2 мы уже рассмотрели один из примеров на анимацию — демонстрация сокращений сердца. На рис. 5.52 показан еще один пример (файл 2link.vsm папки Animate). Этот пример моделирует поведение демпфированного сложного маятника из двух соединенных шарнирно стержней. Верхний стержень также шарнирно закреплен в верхнем конце.

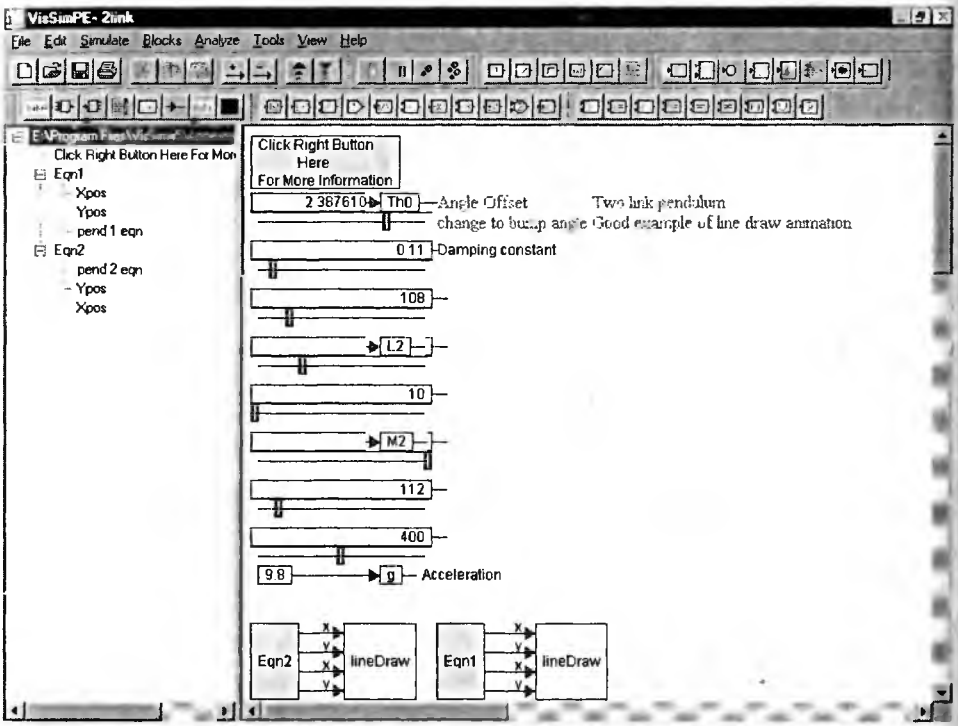


Рис. 5.52. Модель демпфированного маятника из двух стержней

Если отвести систему из двух стержней от состояния равновесия и отпустить стержни, то можно наблюдать довольно сложный колебательный процесс их движения. Он после перехода в режим Display и пуска модели представлен на рис. 5.53 в виде одного из кадров движения маятника.

В этом примере вы можете просмотреть состав его субблоков, которые реализуют необходимые для описания движения каждого стержня. Так, на рис. 5.54 представлена диаграмма блока pend 1 eqn, задающая в виде блочной модели уравнения колебаний первого стержня. В этой диаграмме имеется виртуальный осциллограф, позволяющий контролировать поведение стержня.

Диаграмма другого блока pend 2 eqn, показанная на рис. 5.55, задает в виде блоков уравнения, описывающие колебания второго стержня. Здесь уместно отметить, что колебания стержней взаимосвязаны.

Детальное знакомство с этим примером может быть проведено просмотром диаграмм всех его подмоделей и текстовых комментариев.

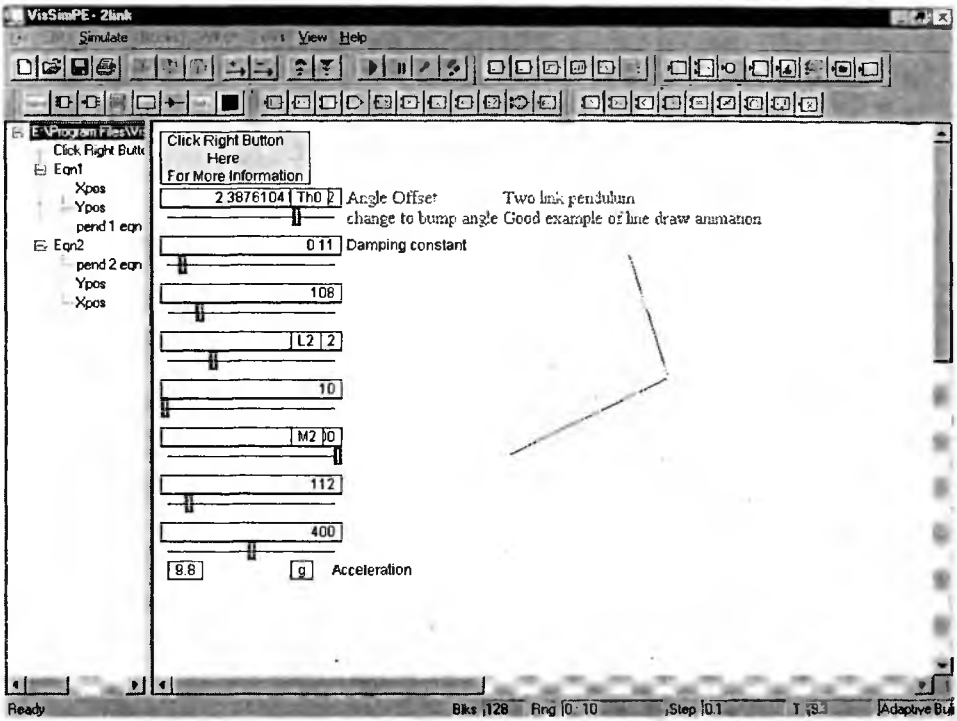


Рис. 5.53. Один из кадров движения маятника из двух стержней, соединенных шарниром

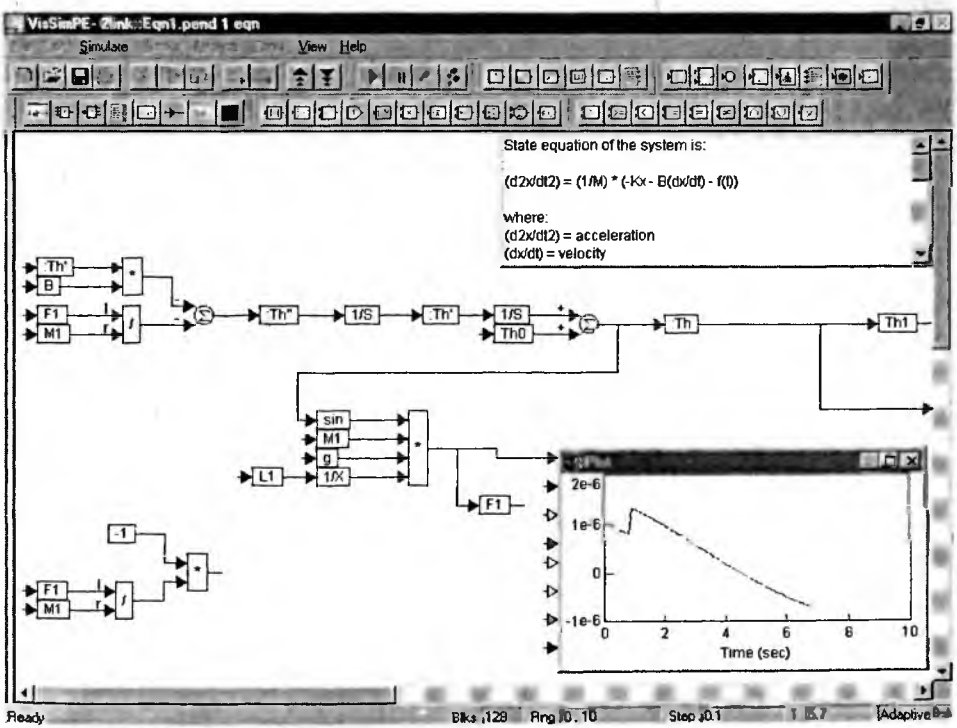


Рис. 5.54. Диаграмма блока pend 1 eqn

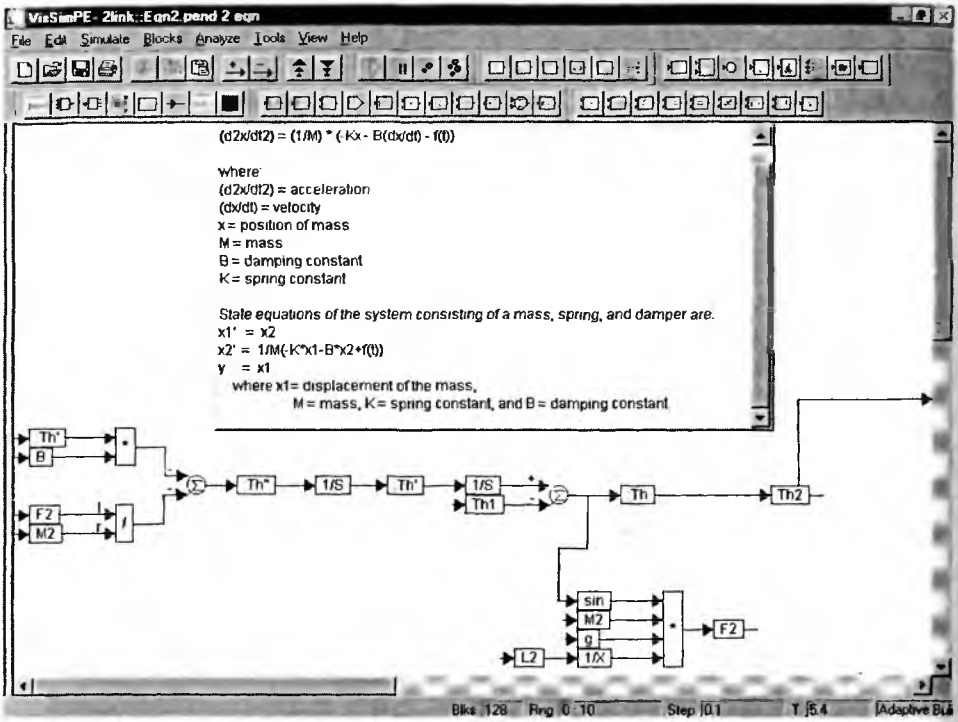


Рис. 5.55. Диаграмма блока pend 2 eqn

5.6.3. Пример на анимацию вращения сферы

В старых версиях Mathcad, поставляемых с системным интегратором MathConnex, многим пользователям запомнился любопытный пример на вращение сферы, составленной из многоугольников. Каждая вершина сферы при таком вращении меняет свои координаты в трехмерном пространстве, и пересчет их выполнялся с помощью матричной системы MATLAB. Но, несмотря на ее хваленную мощь, данный пример запомнился своей крайней медлительностью.

В VisSim этот пример радует глаз быстрым вращением сферы. Модель этого примера представлена на рис. 5.56. В ней видны ползунковые регуляторы для задания начальных установок этого примера и единственный блок для пересчета координат вершин сферы 3D Calculation. Этот блок виден и доступен для просмотра при отказе от режима Display.

Посмотрев в окно дерева диаграммы, можно заметить, что большую часть блоков составляют блоки Line, составленные на основе библиотечного блока Line Draw. И в самом деле, построение сложной фигуры — сферы из многоугольников — в этом примере реализовано просто как построение множества отрезков прямых, координаты концевых точек которых непрерывно пересчитываются в соответствии с работой блока 3D Calculation. Начальные значения координат были вычислены с помощью системы MATLAB, а блок 3D Calculation обеспечивает лишь их пересчет. Такой подход резко уменьшает время вычислений и позволяет сделать вращение сферы более быстрым и плавным.

После перехода в режим Display и пуска модели рис. 5.56 можно наблюдать анимационное вращение сферы. Один из кадров такого вращения пока-

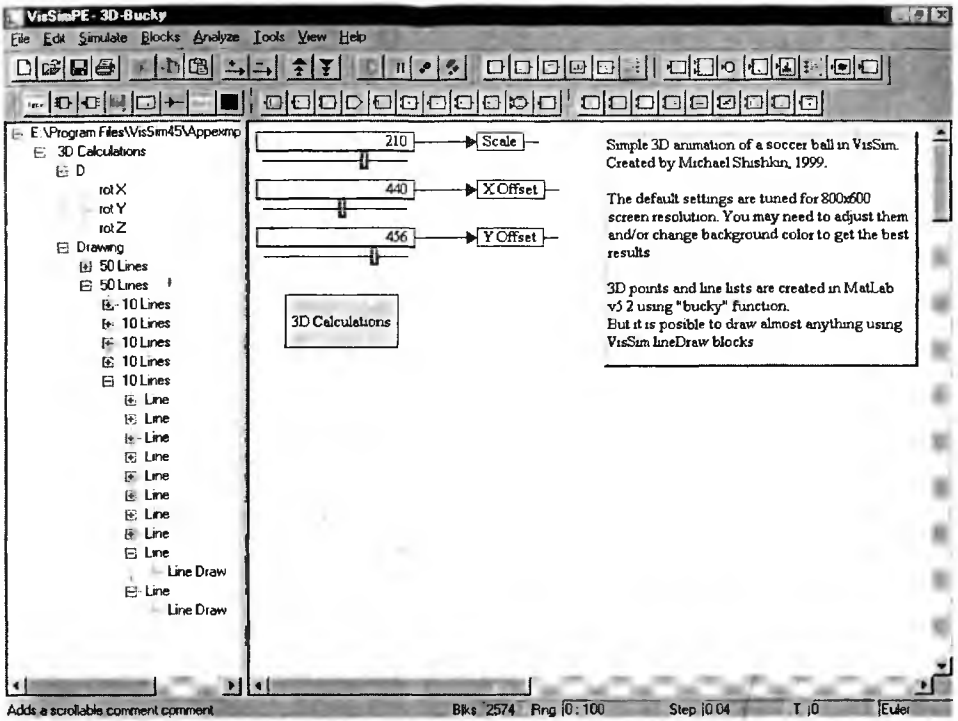


Рис. 5.56. Модель примера 3D-Bucky

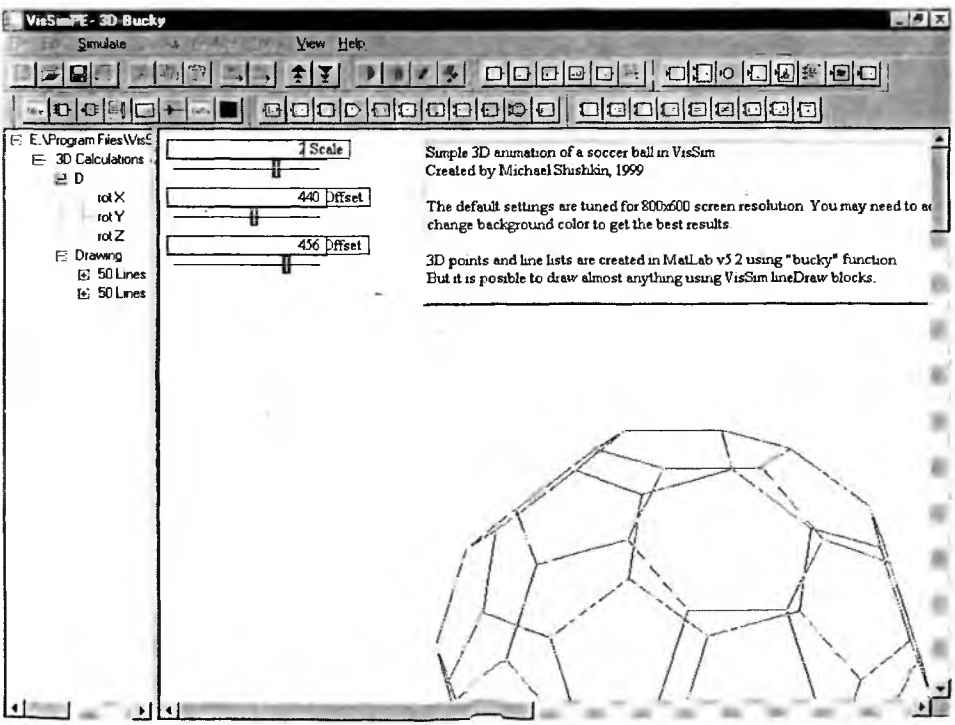


Рис. 5.57. Кадр вращения сферы, составленной из шестиугольных ячеек-плоскостей

зан на рис. 5.57. Здесь видна большая часть сферы и видно, что она составлена из шестиугольных ячеек.

К сожалению, нельзя не отметить громоздкость реализации данного алгоритма и реализующей его модели. Но это тот не слишком частый случай, когда громоздкая модель работает явно более эффективно, чем простая.

5.6.4. Моделирование панели управления с анимацией

С техникой построения анимационных объектов с помощью блока Animation можно познакомиться по примеру, приведенному в файле Pidplate.vsm (рис. 5.58). Этот пример имитирует работу реальной панели управления ПИД-контроллера.

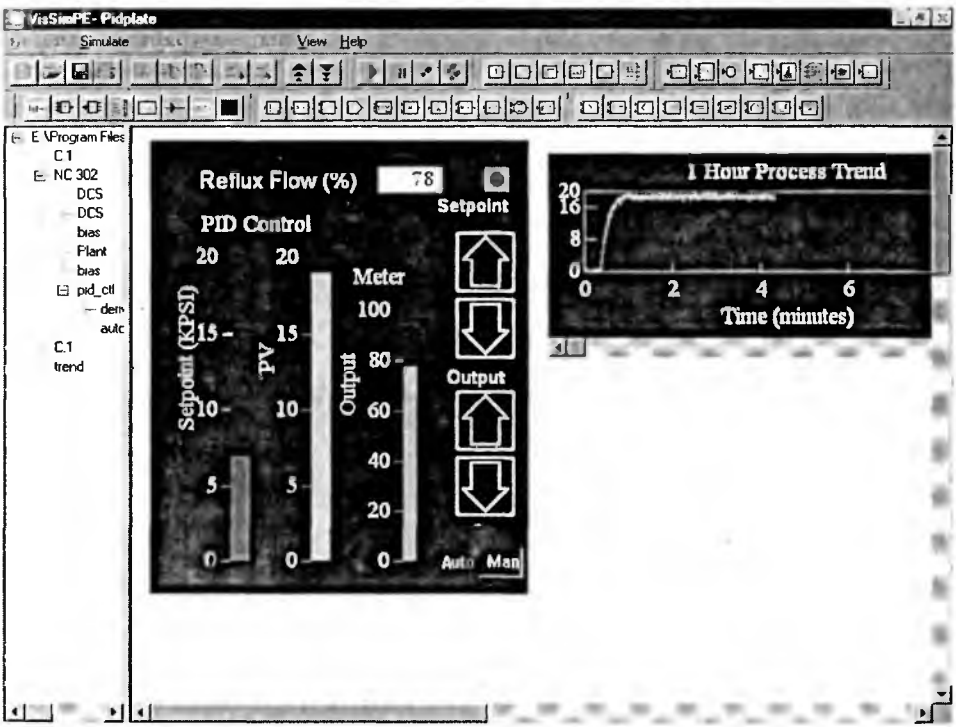


Рис. 5.58. Панель управления ПИД-контроллера

Главным элементом этой панели является «действующий» трехканальный столбцовый индикатор уровня. В примере имитируется его работа — высота столбиков разного цвета меняется в ходе моделирования. Для этого в примере имеются наборы простейших BMP-файлов в виде прямоугольников разной высоты. Они и используются для работы блока Animation.

5.6.5. Моделирование биологического реактора

К классическим моделям нелинейных систем второго порядка относятся модели «хищник—жертва», «питательная среда—бактерии» и т. д. В примере, файл которого называется Bioreact.vsm, задается модель биореактора. Это в

сущности емкость, заполненная бактериями (бактериальной культурой) и питательной средой. Бактерии перерабатывают питательную среду и размножаются. В то же время, прожив некоторое время, они умирают из-за недостатка пищи (питательной среды) и отравления ею и тем самым пополняют питательную среду. В зависимости от функциональной связи между потреблением питательной среды и умиранием бактерий характер изменения концентрации бактерий и питательной среды может быть самым различным — от апериодического до затухающего. Его и позволяет исследовать данная модель.

Модель биореактора представлена на рис. 5.59. Сам биореактор представлен в виде бака с водой, в которой и находятся компоненты питательной среды и бактерии. Ползунковый регулятор позволяет менять уровень дебита воды — основного параметра, от которого зависит характер взаимодействия между бактериальной и питательной средами.

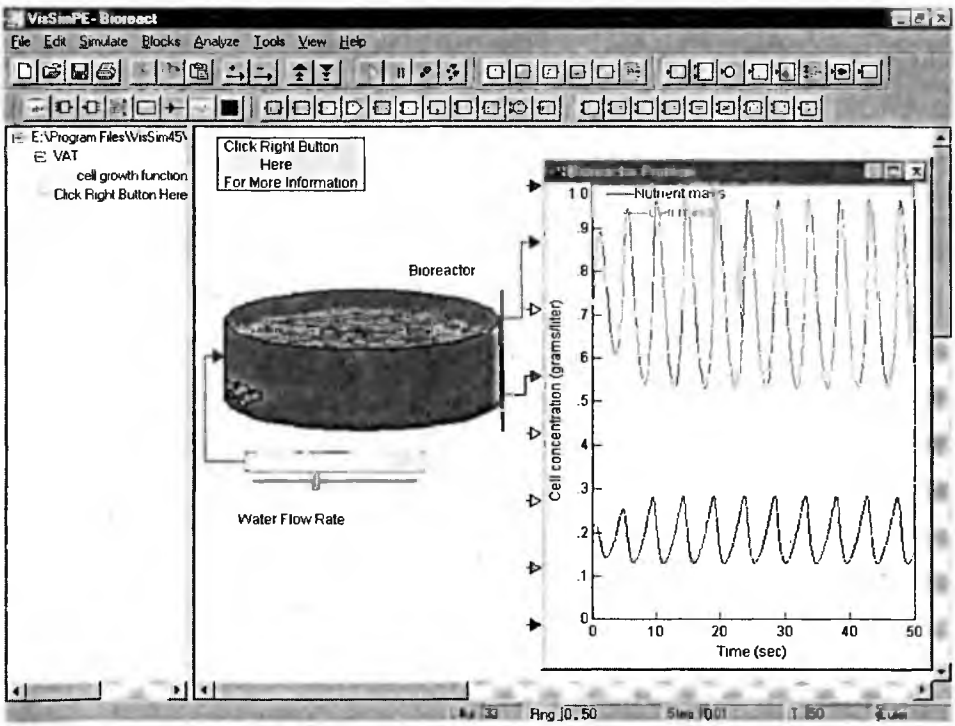


Рис. 5.59. Укрупненная модель биореактора

Активизируя блок биореактора, можно вызвать основной субблок, в котором задается блочная диаграмма биореактора. Она представлена на рис. 5.60 и описывает биореактор как нелинейную систему второго порядка.

Центральным блоков в этой модели является блок функции развития биомассы бактерий — cell growth function. Активизация этого блока открывает окно блочной диаграммы, задающей функцию развития. Она представлена на рис. 5.61.

Таким образом, структура модели биореактора полностью раскрыта. Краткое текстовое описание модели, разумеется, на английском языке имеется в

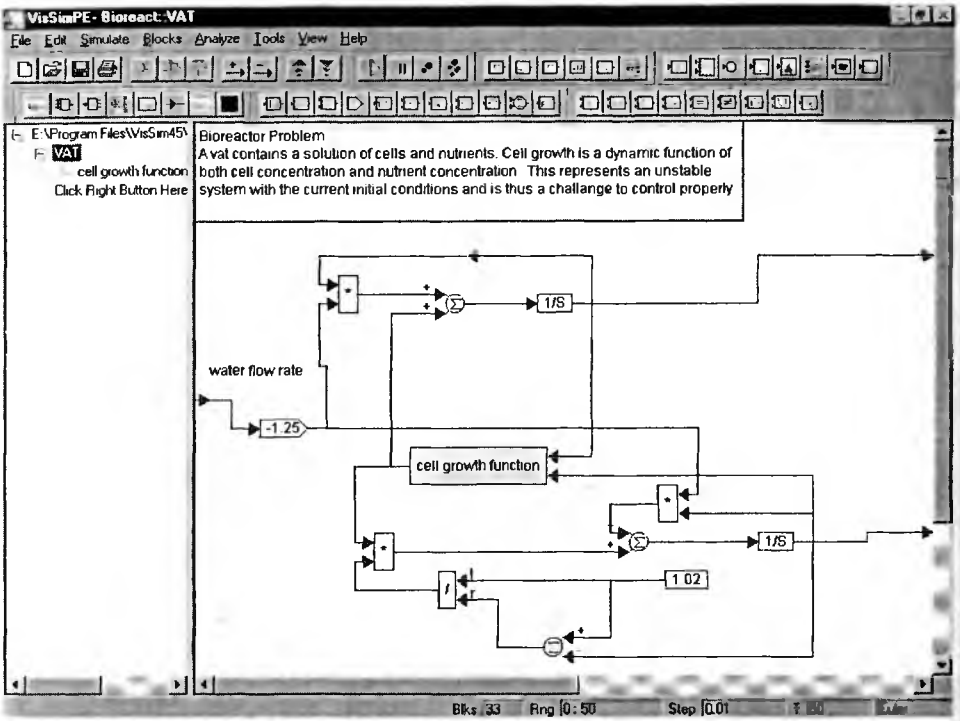


Рис. 5.60. Блочная диаграмма биореактора

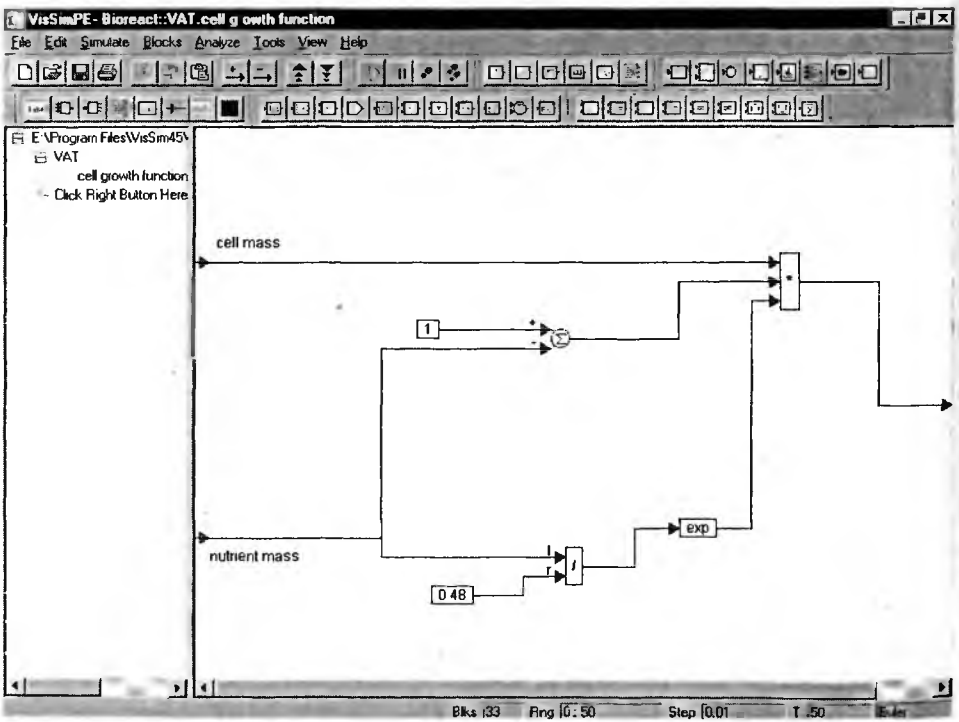


Рис. 5.61. Блочная диаграмма функции развития

блоке Click Right Button Here for More Information (Нажмите правую клавишу мыши для получения дополнительной информации). Большое число подобных задач имеется и в разделе Chemeng (Примеры химической инженерии).

5.7. Средства и примеры моделирования некоторых специальных систем

5.7.1. Блок моделирования гистерезиса

Некоторые устройства (например, сердечники из электромагнитных материалов) и системы обладают гистерезисом (в механических системах именуемым люфтом). Блок Hysteresis Control служит для имитации этого эффекта. Этот блок и пример его применения представлены на рис. 5.62.

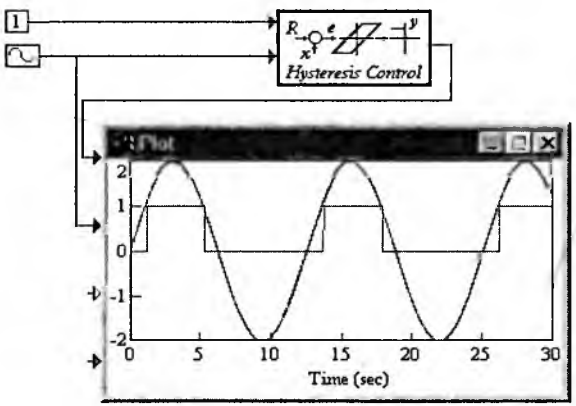


Рис. 5.62. Блок моделирования гистерезиса и его применение

Блок имеет два входа и один выход. На верхний вход подается опорный пороговый сигнал, а на нижний — меняющийся сигнал. Выход у блока один — сигнал равен 0 в пределах петли гистерезиса и 1 за ее пределами.

5.7.2. Блоки реализации логических и управляющих операций

В нелинейных системах довольно часто используются логические и управляющие блоки, использующие условные выражения и циклы. Они позволяют менять структуру модели в зависимости от внешних воздействий или параметров сигналов внутри модели. Это позволяет реализовать сложные системы событийного моделирования. Для облегчения построения таких систем в VisSim 4.5 включен обширный набор таких блоков (рис. 5.63). Они расположены в папке Statchrt, входящей в папку Apprexample демонстрационных примеров.

Эти блоки выполнены как субблоки. При активизации любого из представленных субблоков мышью можно вывести его диаграмму. В качестве примера на рис. 5.64 представлена диаграмма субблока 300 ms min FALSE -block 1. Субблок содержит ряд логических элементов, выполняющих логические операции и функции.

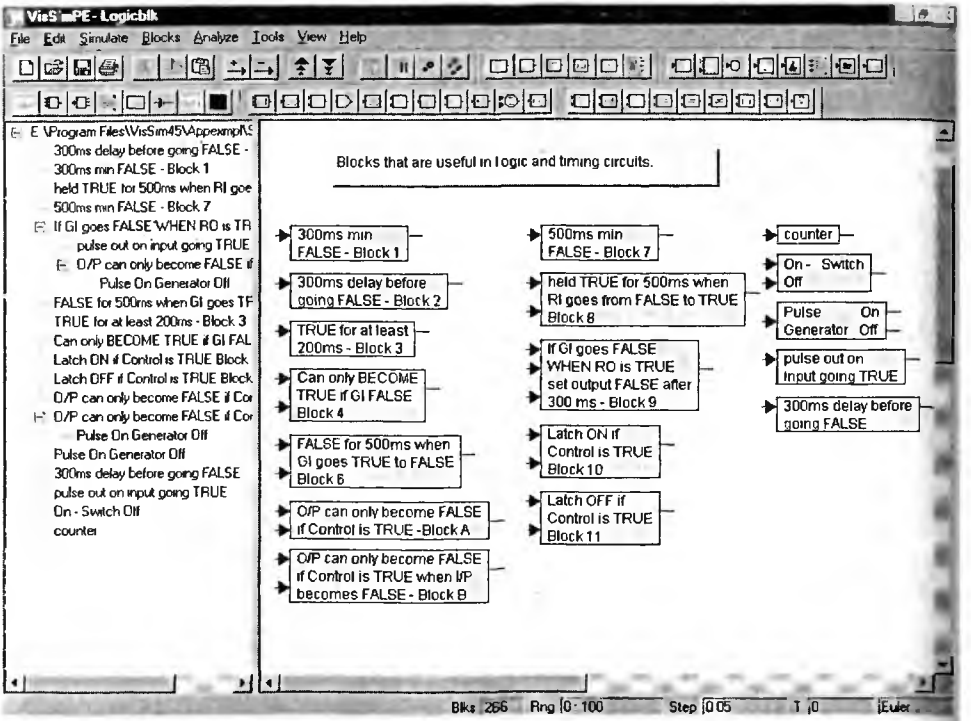


Рис. 5.63. Окно библиотеки логических и управляющих блоков

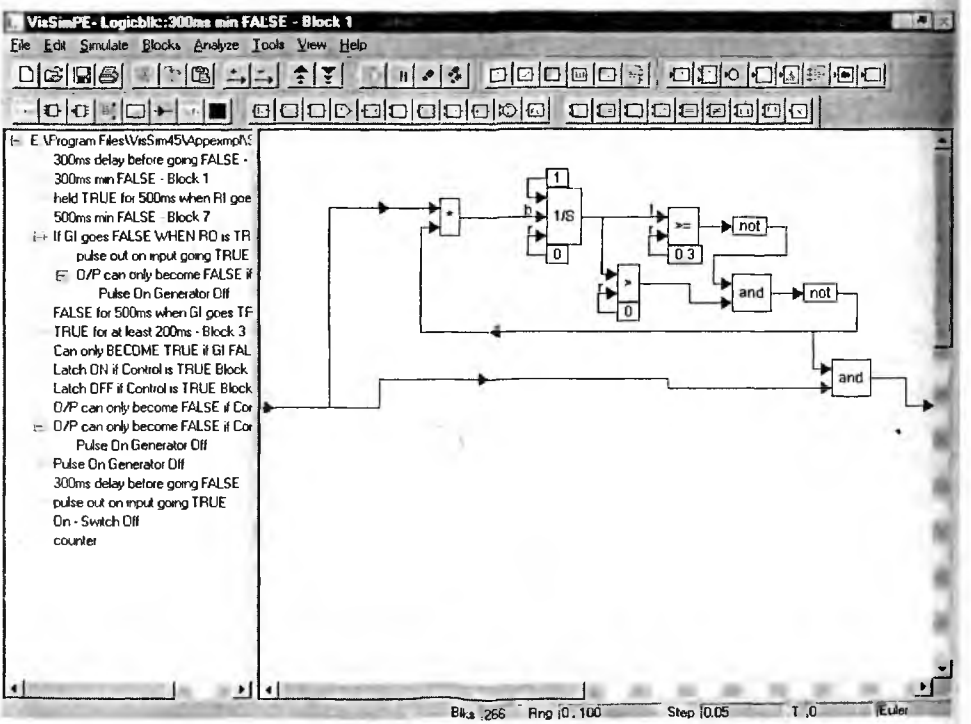


Рис. 5.64. Диаграмма субблока 300 ms min FALSE -block 1

5.7.3. Пример системы с развитым логическим управлением

В указанной папке можно найти также демонстрационный пример построения логически управляемой сложной системы с переменной структурой. Этот пример задан файлом Log_ex1.vsm. Основная модель этого примера показана на рис. 5.65.

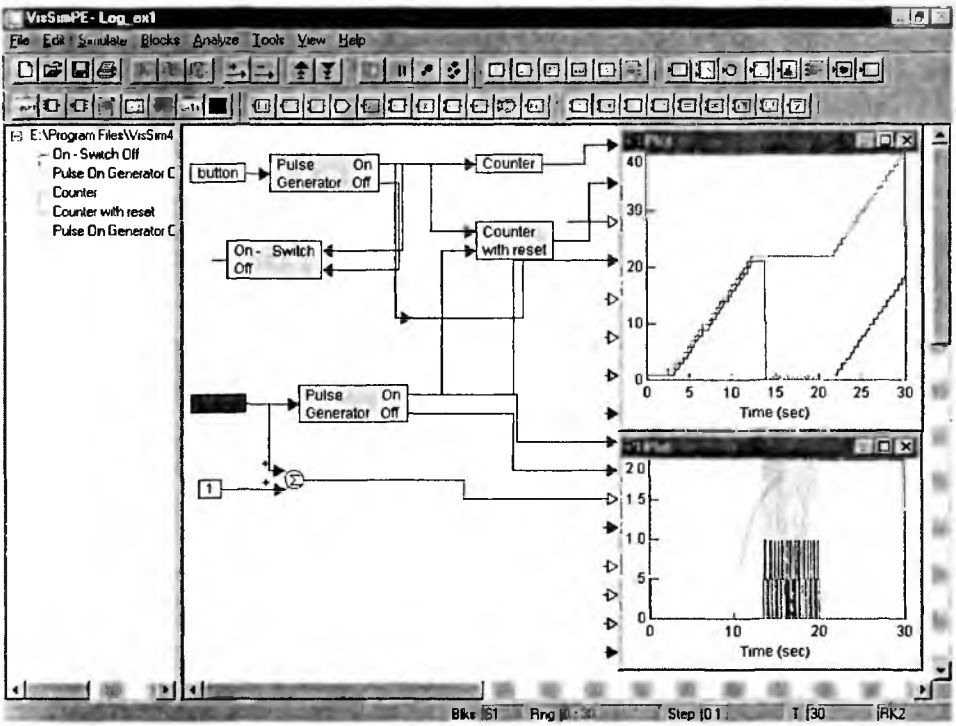


Рис. 5.65. Основная модель логически управляемой системы

С логикой работы данной модели можно ознакомиться, запустив модель на моделирование и управляя вручную блоками button как кнопками. Для нажатия такой кнопки надо разместить курсор мыши на блоке button и нажать правую клавишу мыши. Нажатие/отжатие кнопки контролируется изменением цвета ее фона с белого на красный, и наоборот. Логика работы иллюстрируется осциллограммами. С диаграммами входящих в основную модель блоков можно ознакомиться в обычном порядке, активизируя блоки мышью. Логика работы этой модели достаточно сложна, и заинтересованный в изучении подобных систем пользователь может изучить ее в деталях самостоятельно.

5.7.4. Пример системы управления баком с жидкостью

К сложным системам относятся системы управления баками с жидкостью, имеющими входные и выходные клапаны со случайными законами управления. VisSim имеет большой набор примеров моделирования таких систем. Упомянем лишь один из них, модель которого представлена на рис. 5.66. Это модель управления баком с PID-контроллером.

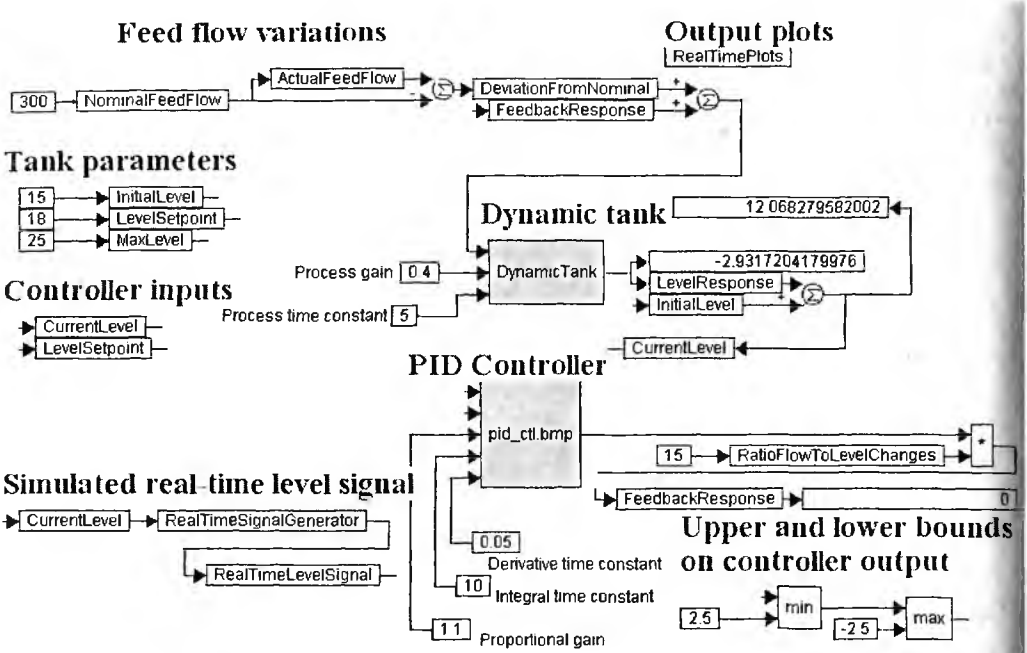


Рис. 5.66. Модель управления баком с PID-контроллером

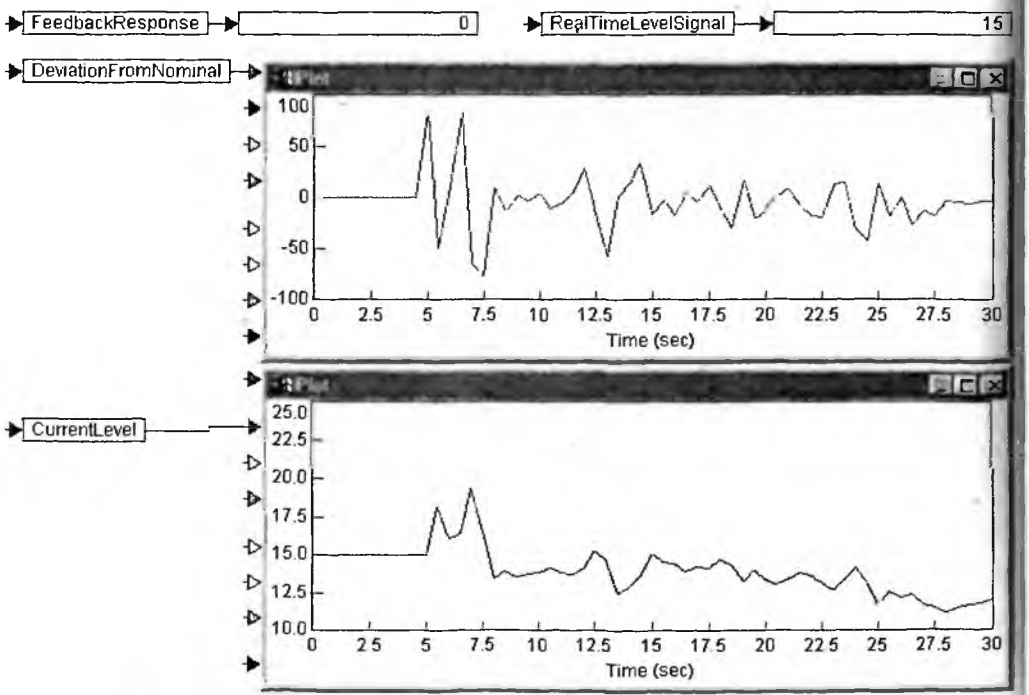


Рис. 5.67. Временные диаграммы работы бака

Модель обеспечивает реализацию метода Монте-Карло, т. е. задание ряда параметров, задающих уровень жидкости в баке с помощью случайных чисел. Поддержание уровня жидкости в допустимых пределах обеспечивает PID-контроллер, свойства которого уже описывались в главе 4. О сложности процессов в системе свидетельствуют временные диаграммы работы бака, показанные на рис. 5.67.

Этот пример показывает, что реальные системы управления баками с жидкостью гораздо сложнее такой простой системы, как сливной бачок в туалете. Не случайно в VisSim и включено множество примеров на построение систем управления баками и построения гидравлических систем. Их детальное описание выходит за рамки данной книги.

5.8. Моделирование электротехнических и электромеханических систем

5.8.1. Трудности моделирования электротехнических устройств

Электротехнические и электромеханические системы — благодатная область применения VisSim, хотя она специально для моделирования таких систем не предназначена. Это, кстати, создает определенные проблемы в ее применении.

Трудности заключаются в отсутствии в библиотеке компонентов блоков моделей даже таких простейших устройств, как резисторы R, конденсаторы C и катушки индуктивности L. Впрочем, их легко создать на основе известных соотношений между напряжениями на этих элементах и токами, текущими через них:

$$u(t) = R \cdot i(t), \quad i(t) = C\frac{du(t)}{dt}, \quad u(t) = L \cdot \frac{di(t)}{dt}.$$

Как нетрудно заметить, моделирование этих устройств требует применения блоков умножения и вычисления производных. Однако создание модели и уравнений ее работы не автоматизировано и их приходится делать «вручную». В этом отношении программы схемотехнического моделирования Pspice, Design Lab, MicroCAP и Electronic Workbench оказываются намного удобнее, поскольку имеют библиотеки блоков, содержащие не только резисторы, конденсаторы и катушки индуктивностей, но и такие компоненты как диоды, транзисторы, тиристоры, линии задержки, трансформаторы и т. д.

Однако ситуация начинает меняться, когда необходимо провести моделирование сложных электрических и электромеханических устройств, в состав которых входят электродвигатели, регуляторы напряжения и тока и иные устройства, функционирование которых задается на уровне макромоделей. Для этого средства VisSim оказываются вполне пригодными.

5.8.2. Моделирование двухполупериодного выпрямителя

Модель двухполупериодного выпрямителя с конденсаторным фильтром (рис. 5.68) является наглядным примером трудности подготовки модели простого электротехнического устройства. Изображение самого выпрямителя, с почему-то встречно включенными диодами, выглядит несколько странно, правильнее было бы показать обычную трансформаторную схему выпрямителя со средней точкой.

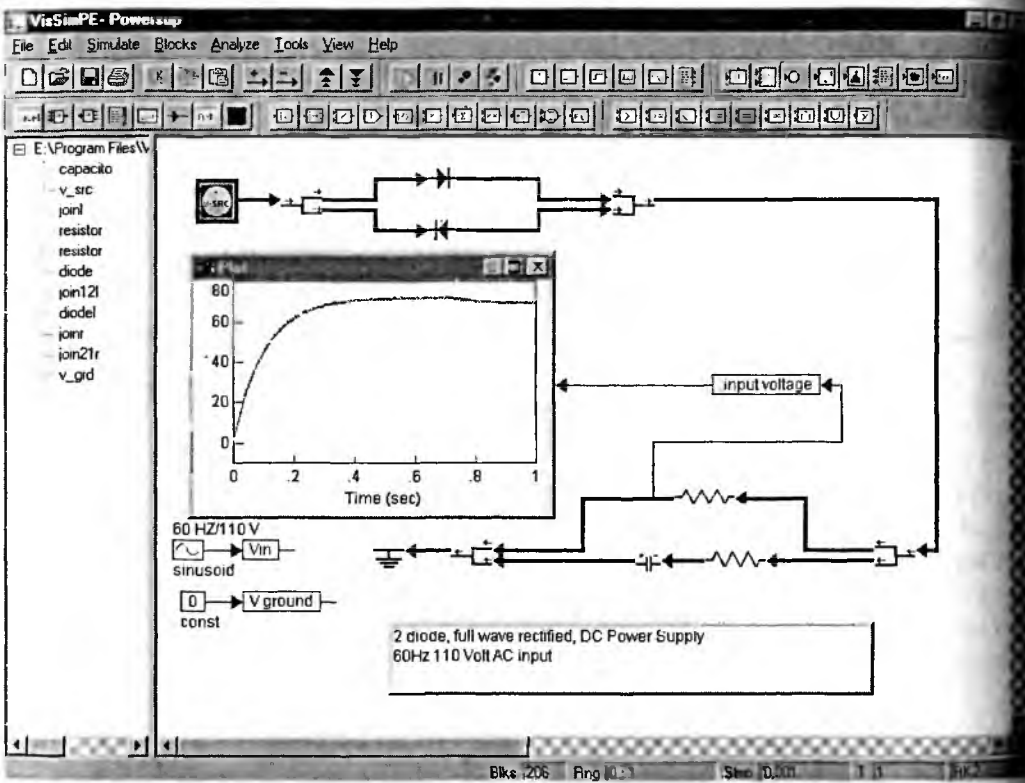


Рис. 5.68. Модель двухполупериодного выпрямителя

В левой части экрана видно дерево этой модели, в которое входят субблоки конденсатора фильтра, диодов, резисторов и др. Ограничимся приведением диаграмм субблоков диода и конденсатора фильтра. Диаграмма модели диода представлена на рис. 5.69. Диод в обратном включении представлен большим сопротивлением backward resistance. В прямом включении диод закрыт до порога voltage thereshold = 0,7 В (на это показывает цифровой индикатор), выше порога диод представляется малым сопротивлением forward resistance. Таким образом, используется широко распространенная упрощенная кусочно-линейная аппроксимация вольт-амперной характеристики диода. Осциллограф на рис. 5.69 показывает временную зависимость напряжения однофазной сети переменного тока (110 В, 60 Гц) и временную зависимость напряжения на выходе выпрямителя (диодов).

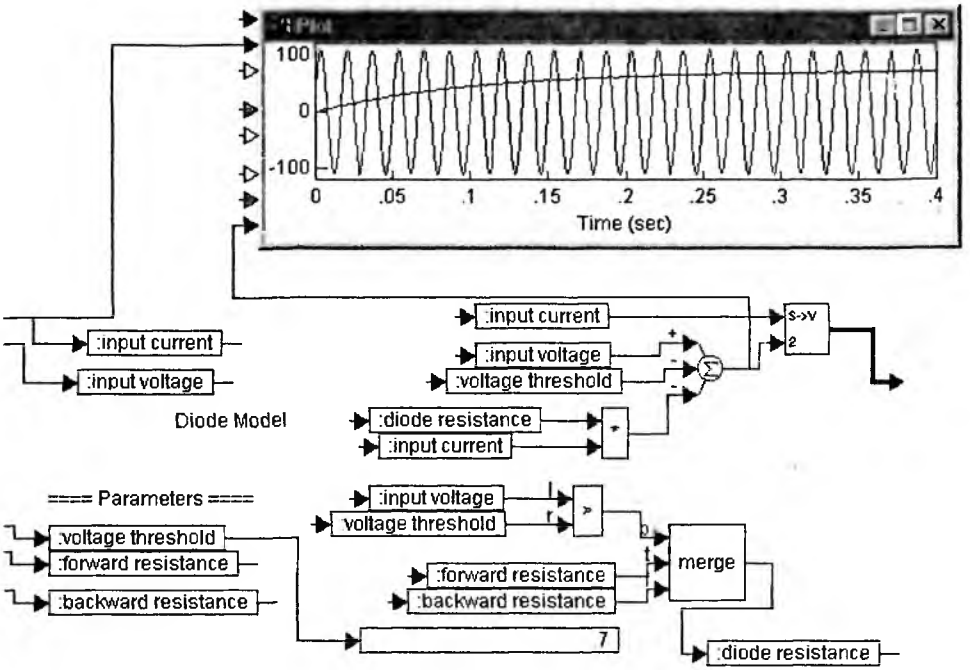


Рис. 5.69. Диаграмма модели субблока диода

Диаграмма модели конденсатора фильтра представлена на рис. 5.70. Нетрудно заметить, что она реализует приведенное выше соотношение для i(t), причем дифференцирующее устройство реализовано на базе блока интегратора. Диа-

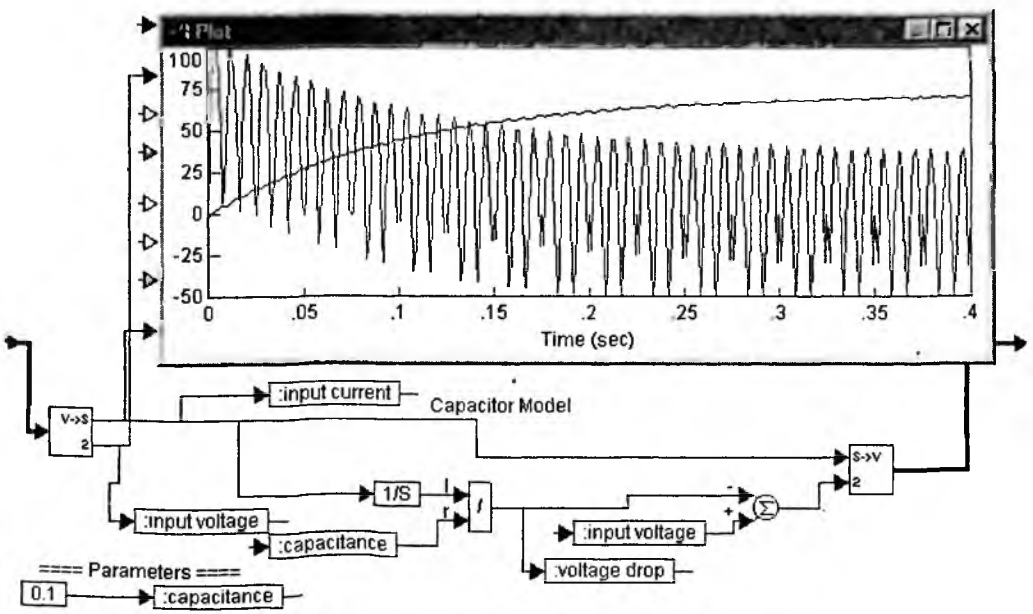


Рис. 5.70. Диаграмма субблока конденсатора фильтра

грамма дополнена блоком осциллографа, который позволяет просмотреть временную диаграмму тока в цепи конденсатора и напряжения на нем. Отчетливо видна начальная стадия заряда конденсатора, соответствующая нарастанию напряжения на выходе выпрямителя. Ток через конденсатор пульсирующий, поскольку заряд идет полуволнами входного синусоидального сигнала (110 В 60 Гц). В установившемся режиме конденсатор то заряжается, то разряжается, отдавая энергию в промежутке между пиками выпрямленного напряжения.

Итак, результаты моделирования вполне отвечают представлениям о работе двухполупериодного выпрямителя с конденсаторным фильтром на активную нагрузку. Как уже отмечалось, модель выпрямителя в целом переусложнена необходимостью создания субблоков ее компонентов.

5.8.2. Моделирование двигателя постоянного тока

А теперь рассмотрим достаточно сложный объект моделирования — двигатель постоянного тока. Основная модель двигателя представлена на рис. 5.71. Собственно говоря, это даже не модель двигателя, а простая схема его испытания и сравнения данных моделирования с экспериментом. Двигатель (субблок Motor 12B) запускается источником перепада, и вычисляется временная зависимость скорости его вращения. Она сравнивается с экспериментальной зависимостью, которая хранится в блоке Motor 12B Data.

Между тем в данном примере использована одна из самых простых моделей двигателя постоянного тока, представленная на рис. 5.72. Нетрудно заметить, что модель соответствует системе с обратной связью и интегратором, учитывающим инерционные свойства двигателя.

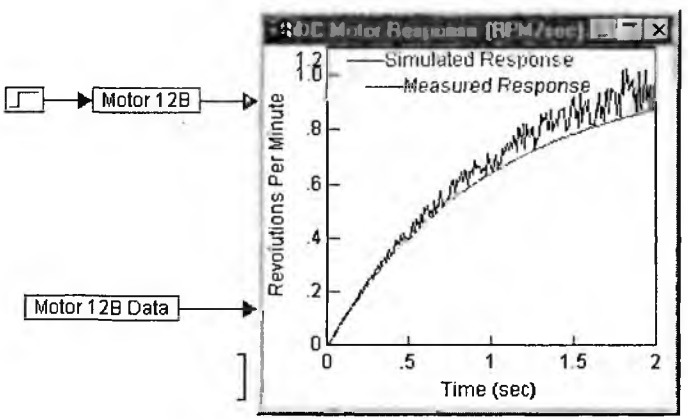


Рис. 5.71. Основная модель испытания двигателя постоянного тока

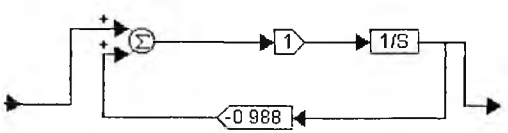


Рис. 5.72. Модель (субблок Motor 12B) двигателя постоянного тока

Нетрудно заметить, что реальная зависимость скорости вращения двигателя от времени отличается от полученной при моделировании заметной зашумленностью данных. Это следствие как случайных погрешностей измерения скорости вращения, так и присущей двигателям постоянного тока хаотическим изменениям скорости. Если не учитывать «шум» экспериментальной кривой, то можно сделать вывод, что законы нарастания скорости вращения для двигателя постоянного тока, полученные при моделировании и при эксперименте, практически идентичны (качественно полностью, а количественно с вполне приемлемой погрешностью).

5.8.3. Моделирование двухмоторной системы

Следующий пример, который мы рассмотрим, — работа двух моторов на общую нагрузку. Два разных мотора (двигателя постоянного тока) подключены к нагрузке через эластичные ременные передачи. Как будет распределяться потребляемый моторами ток, изменяться во времени скорость и угол поворота? Эту достаточно сложную задачу решает модель рис. 5.73, представленная в демонстрационном примере 2dcmots.vsm.

Эта модель достаточно сложна. В этом можно убедиться, рассмотрев диаграмму главного субблока i1 V1 w1 theta 1 i2 V2 w2 theta 2. Этот основной блок, диаграмма которого дана на рис. 5.74, вычисляет параметры (ток, напряжение, частоту и угол) для первого мотора.

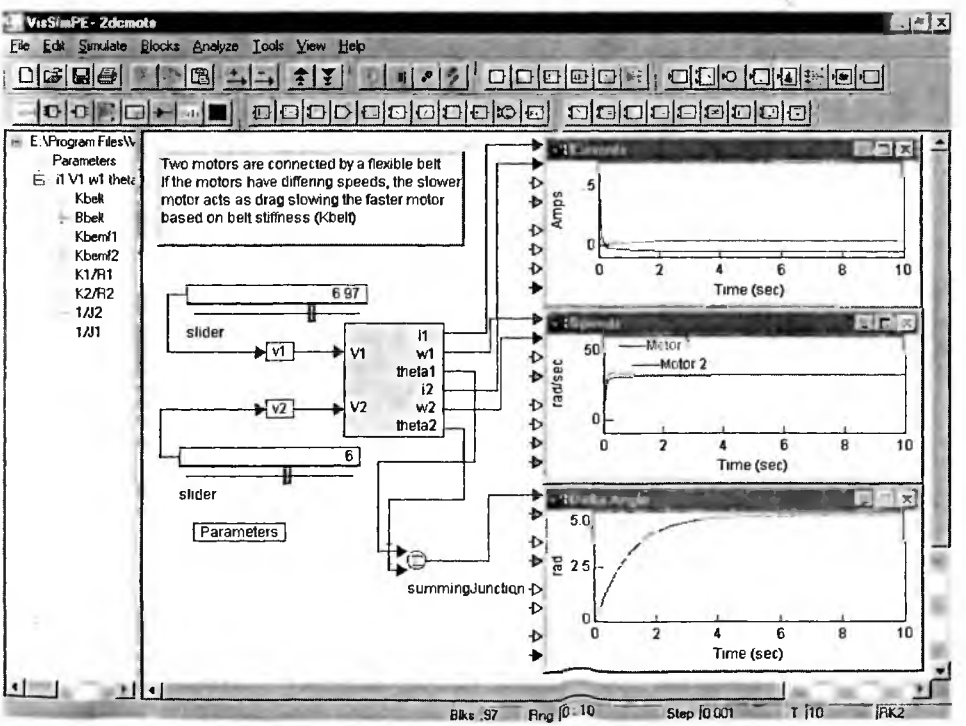


Рис. 5.73. Основная модель двухмоторной системы

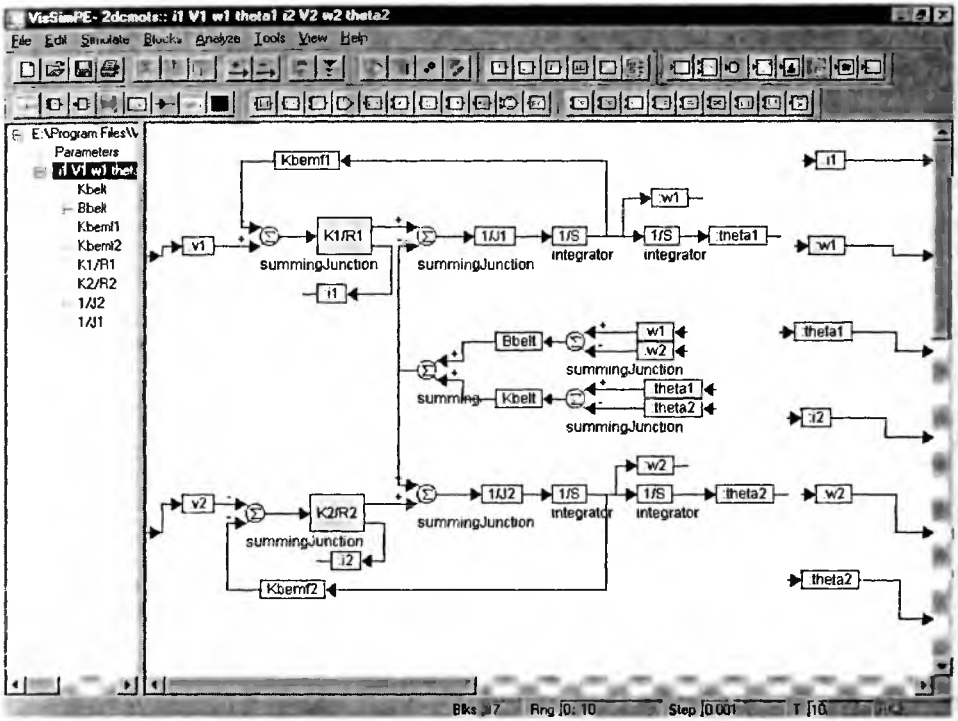


Рис. 5.74. Диаграмма модели субблоков i1 V1 w1 theta 1 i2 V2 w2 theta 2

С остальными субблоками заинтересованный читатель может ознакомиться самостоятельно (назначение субблоков вполне очевидно из их названий). Осциллографы на рис. 5.74 позволяют оценить динамику работы двухмоторной системы — изменение во времени потребляемого моторами тока, скорости вращения и возникшего фазового сдвига.

5.8.4. Моделирование разгона трехфазного асинхронного двигателя

В отличие от двигателей постоянного тока асинхронные двигатели не имеют коллектора. Они более неприхотливы в эксплуатации и широко применяются в промышленности. Особенно это относится к асинхронным двигателям, получающим питание от трехфазной сети.

Однако поведение асинхронного двигателя при разгоне разительно отличается от такового для двигателя постоянного тока. Крутящий момент асинхронного двигателя при низких скоростях вращения содержит высокочастотную составляющую, частота которой определяется частотой сети переменного тока, от которой питается двигатель. Эта составляющая постепенно уменьшается и сводится почти к нулю, когда двигатель входит в режим, граничащий с синхронным. Теоретическое рассмотрение поведения асинхронного двигателя далеко выходит за рамки тематики данной книги, поэтому мы остановимся на формальном описании модели разгона такого двигателя, представленной в демонстрационном файле Acmotor.vsm. Основная модель разгона асинхронного двигателя показана на рис. 5.75.

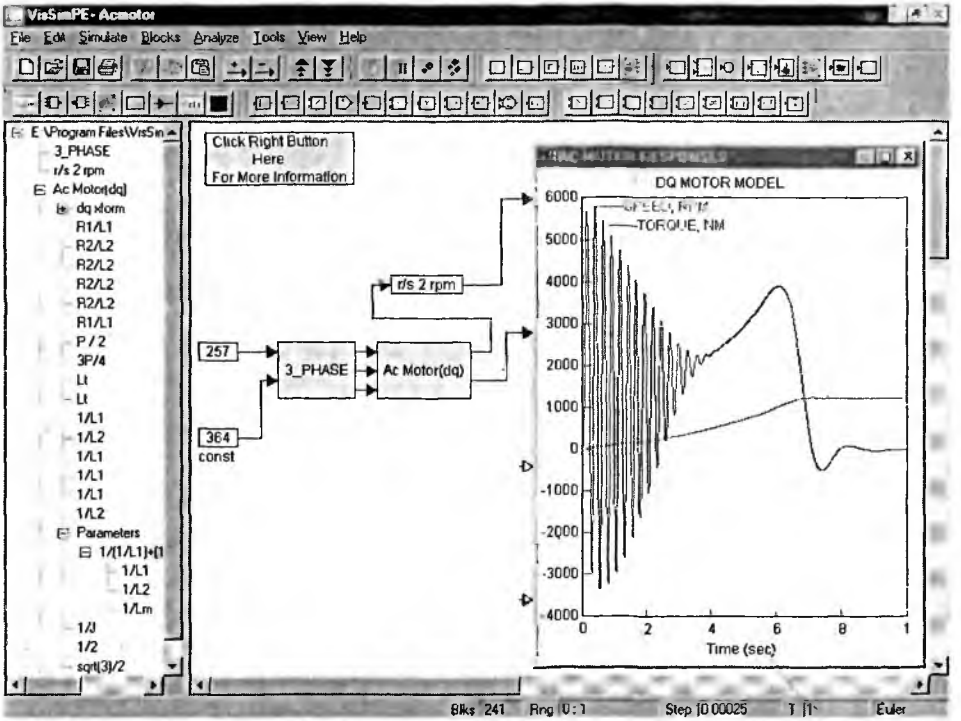


Рис. 5.75. Основная модель разгона асинхронного двигателя.

На рис. 5.75 особенно впечатляюще выглядят осциллограммы крутящего момента и динамики изменения во времени скорости вращения двигателя. Осциллограмма крутящего момента хорошо иллюстрирует сказанное о динамике разгона асинхронного двигателя.

На основной модели представлено всего три главных субблока. Как обычно, доступ к их диаграммам осуществляется активизацией рисунка субблока мышью. Диаграмма субблока r/s 2 rpm, выполняющего тривиальный пересчет частоты из рад/с в герцы, представлен на рис. 5.76.

Субблок задания трехфазного напряжения (модель трехфазной промышленной сети переменного тока) показан на рис. 5.77. Этот субблок в особом описании не нуждается. Можно лишь отметить, что такое напряжение создается тремя каналами с фазовыми сдвигами, соответственно равными 0, $2\pi/3$ и $4\pi/3$.

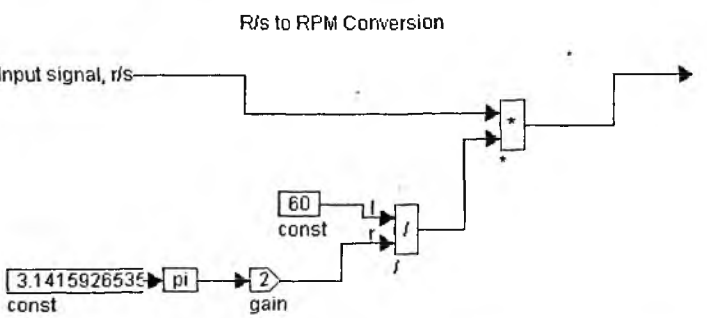


Рис. 5.76. Диаграмма субблока r/s 2 rpm

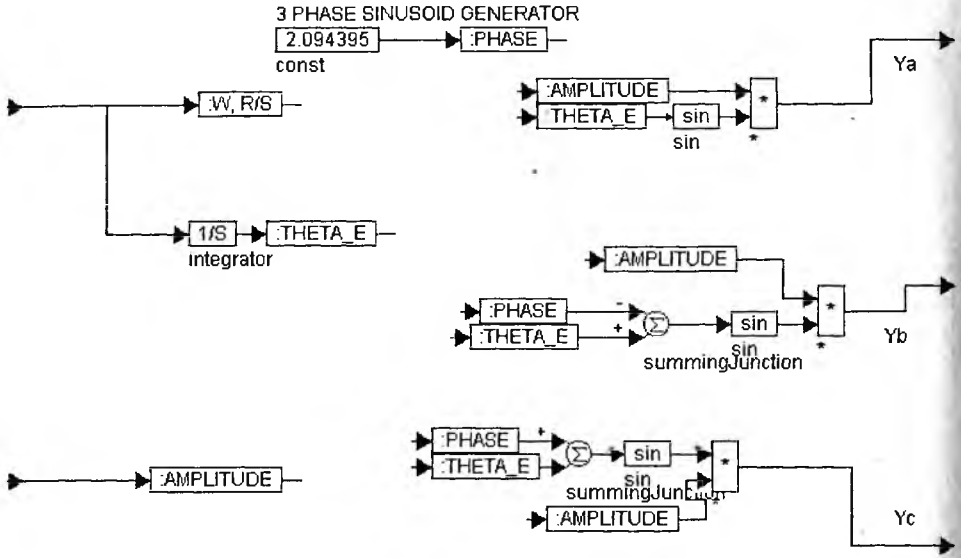


Рис. 5.77. Диаграмма субблока моделирования трехфазной сети

Наконец, последний субблок AC motor (dq), диаграмма которого представлена на рис. 5.78, по существу, и представляет полную модель трехфазного асинхронного двигателя. Это довольно сложная модель, понятная только специалистам в области электропривода.

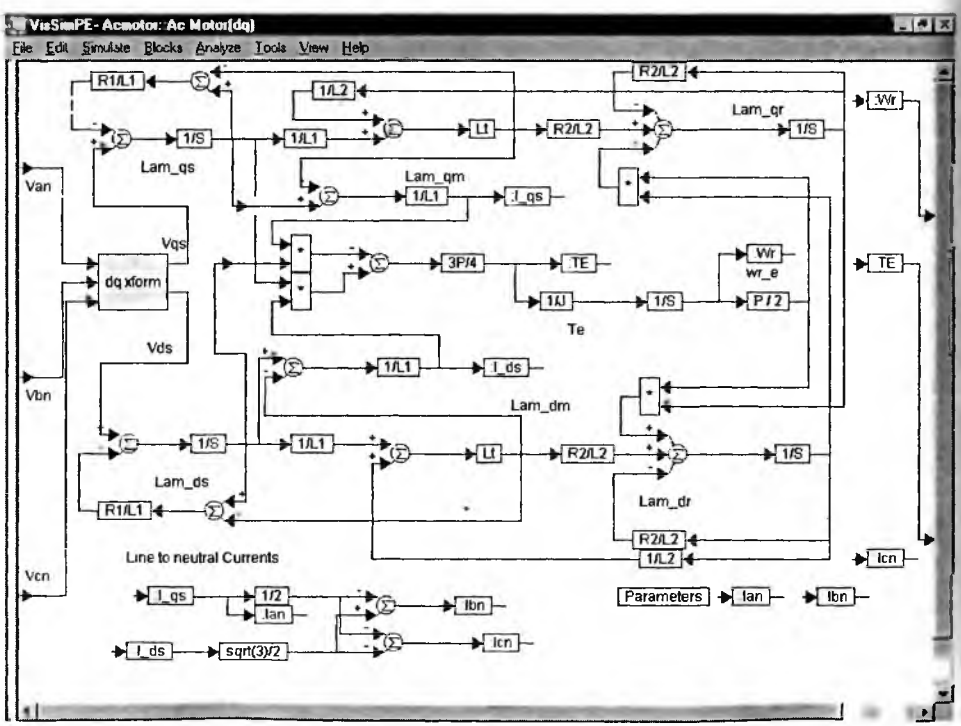


Рис. 5.78. Диаграмма субблока асинхронного двигателя

5.8.5. Моделирование электропривода коробки передач

Примером моделирования нелинейной электромеханической системы является модель электропривода автоматической коробки передач от электродвигателя постоянного тока. Основная модель этой системы показана на рис. 5.79. Обращает на себя внимание оформление постановки задачи в виде рисунка, дающего детальное представление о сути устройств, входящих в систему. Показаны двигатель постоянного тока, редуктор и коробка передач.

Это типичный пример нелинейной системы с гистерезисом. Он позволяет мотору быстро набрать скорость перед переключением коробки на очередную скорость. Наличие гистерезиса типа зон нечувствительности отчетливо видно на осциллограмме, приведенной на рис. 5.79.

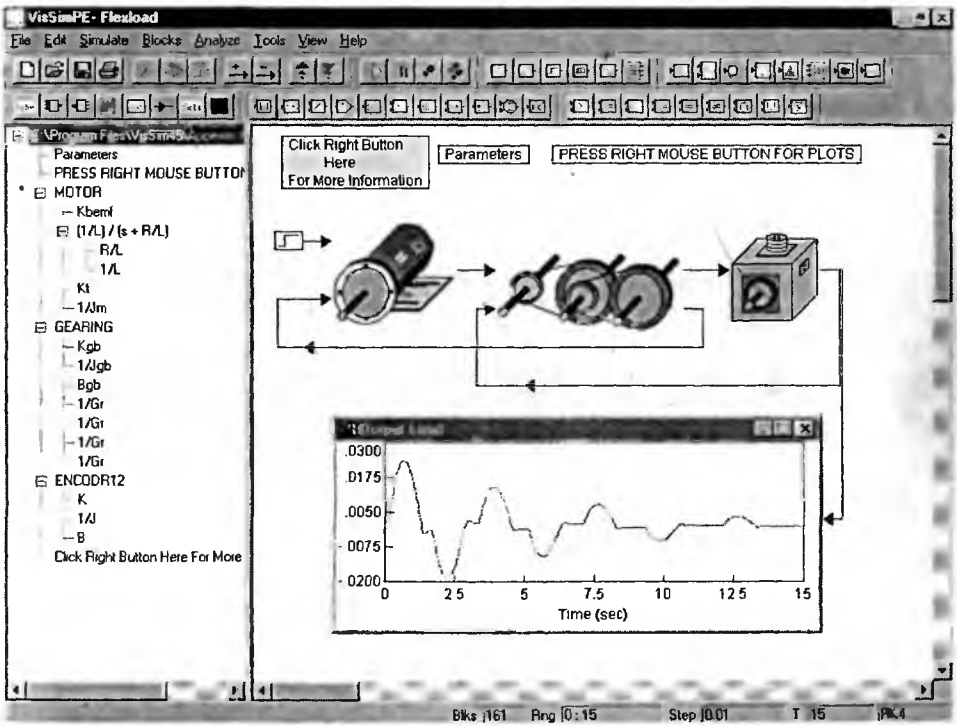


Рис. 5.79. Основная модель электропривода коробки передач

Основная модель имеет субблок задания исходных параметров Parameters. Его активизация открывает диаграмму блока, показанную на рис. 5.80.

Блок с вполне понятным назначением PRESS RIGHT MOUSE BUTTON FOR PLOTS (Нажмите правую клавишу мыши для получения графиков) выводит детальные графики сложных переходных процессов в этой модели. Они показаны на рис. 5.81

Специфика моделирования данной системы, безусловно, связана с моделью коробки передач, имеющей блок нечувствительности Deadband. Диаграмма субблока коробки передач представлена на рис. 5.82.

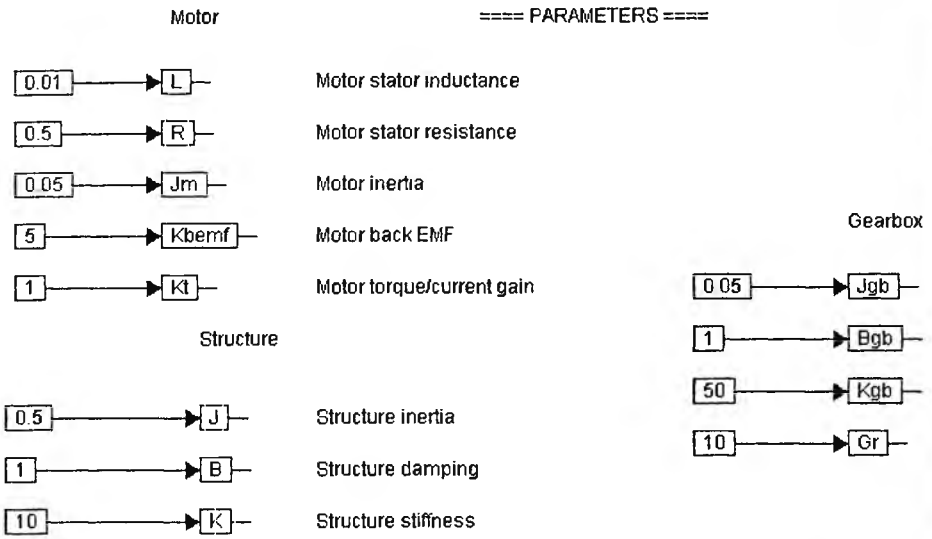


Рис. 5.80. Диаграмма субблока параметров

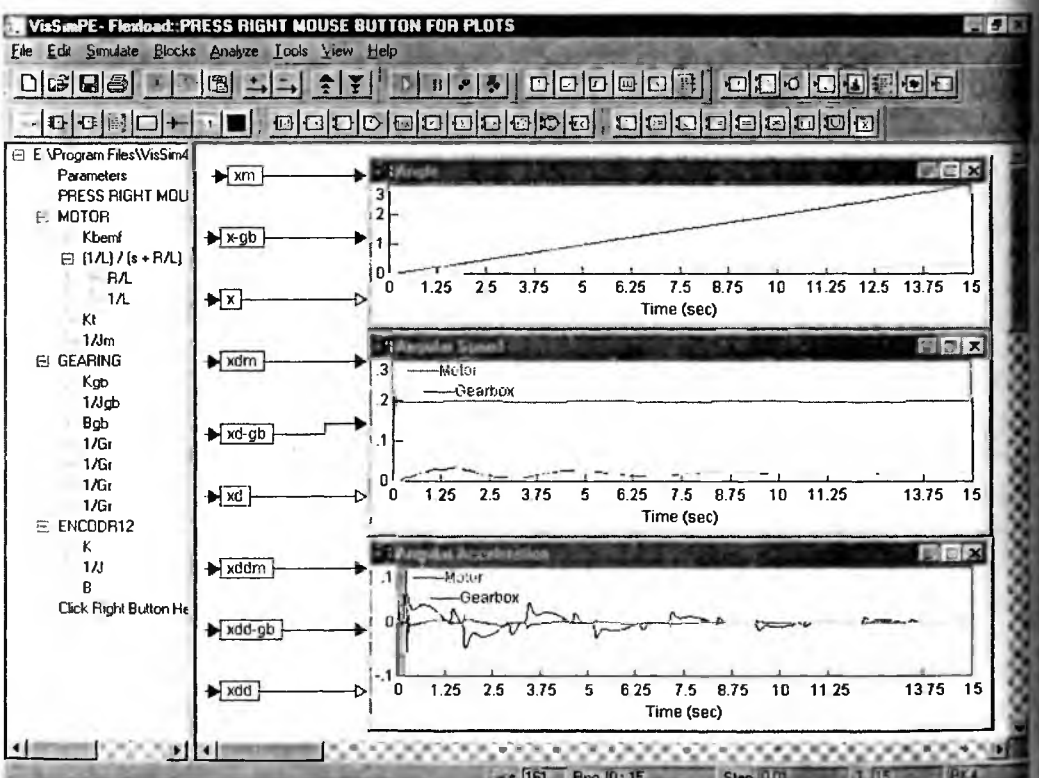


Рис. 5.81. Субблок вывода графиков переходных процессов

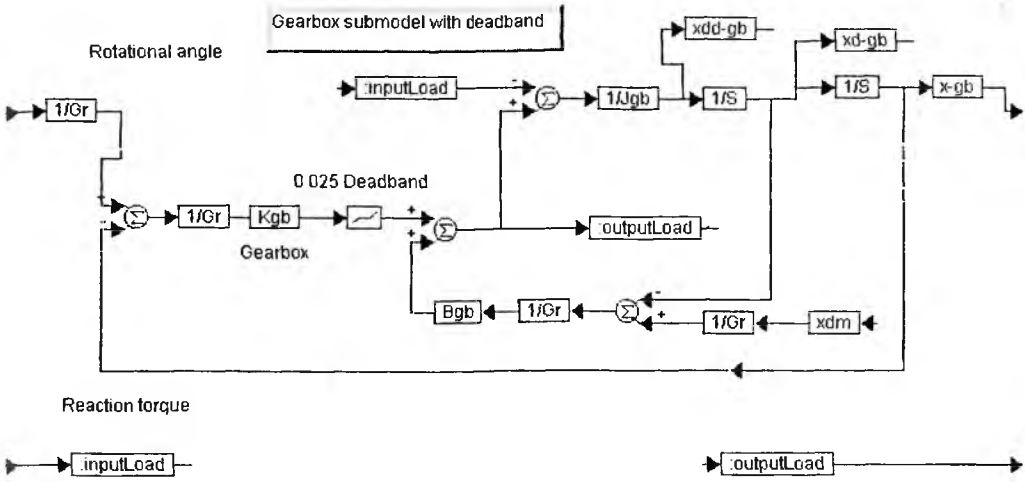


Рис. 5.82. Диаграмма субблока модели коробки передач

5.8.6. Моделирование электромеханической дверной системы

В заключение этого раздела рассмотрим пример на моделирование электромеханической системы открытия и закрытия дверей. Эта система (файл Doorsys.vsm) является хорошим примером оформления модели в системе VisSim. Основной блок модели выглядит достаточно просто (рис. 5.83). Он имеет суб-

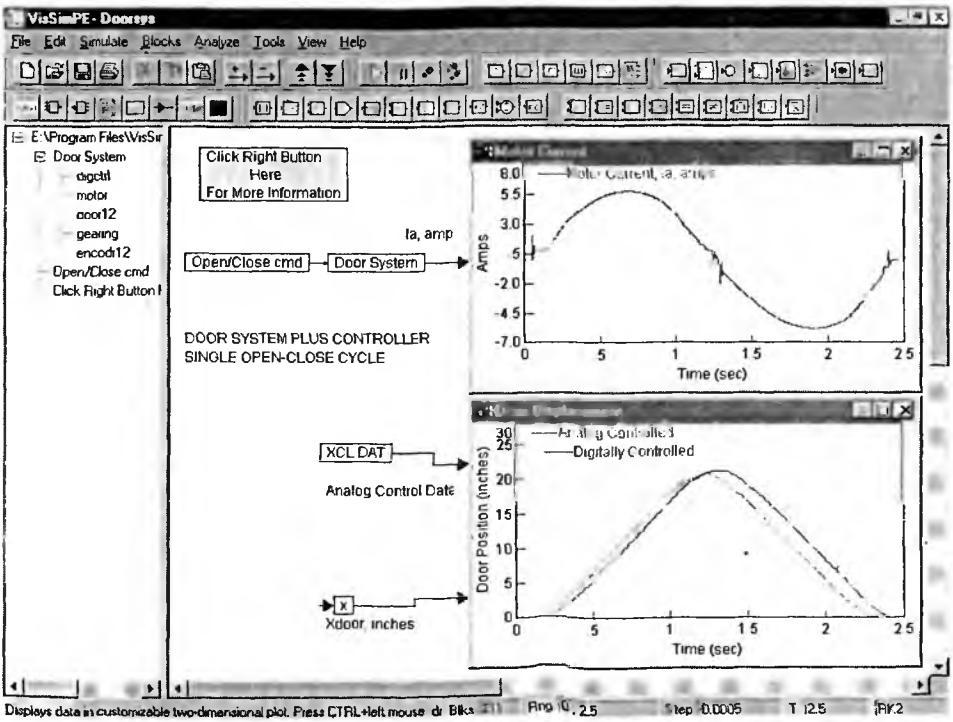


Рис. 5.83. Основной блок модели дверной системы

блоки вызова справки нажатием правой клавиши мыши, блок логики и параметров Open/Close cmd и основной субблок модели Door System.

Верхний осциллограф представляет временную зависимость потребляемого двигателем тока в процессе открывания, а затем закрывания двери. В системе применяется реверсивный двигатель постоянного тока, позволяющий менять направление вращения — при положительном направлении тока дверь открывается, а при отрицательном — закрывается. Временные зависимости тока свидетельствуют о наличии моментов, когда двигателю приходится преодолевать инерцию двери.

Нижний осциллограф дает сравнение двух реализаций дверной системы — на основе цифрового и аналогового регуляторов. Аналоговый регулятор в этом примере не моделируется — для отображения его временной зависимости берутся из файла.

Субблок логики представлен на рис. 5.84.

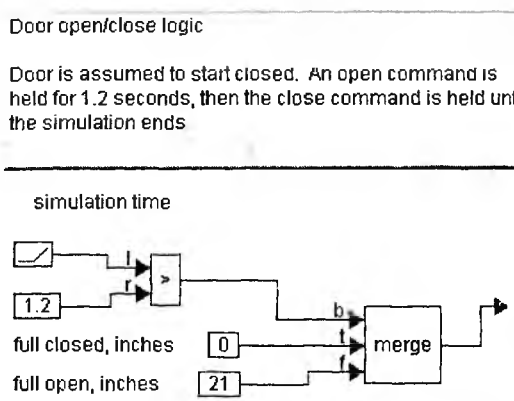


Рис. 5.84. Субблок логики и задания параметров

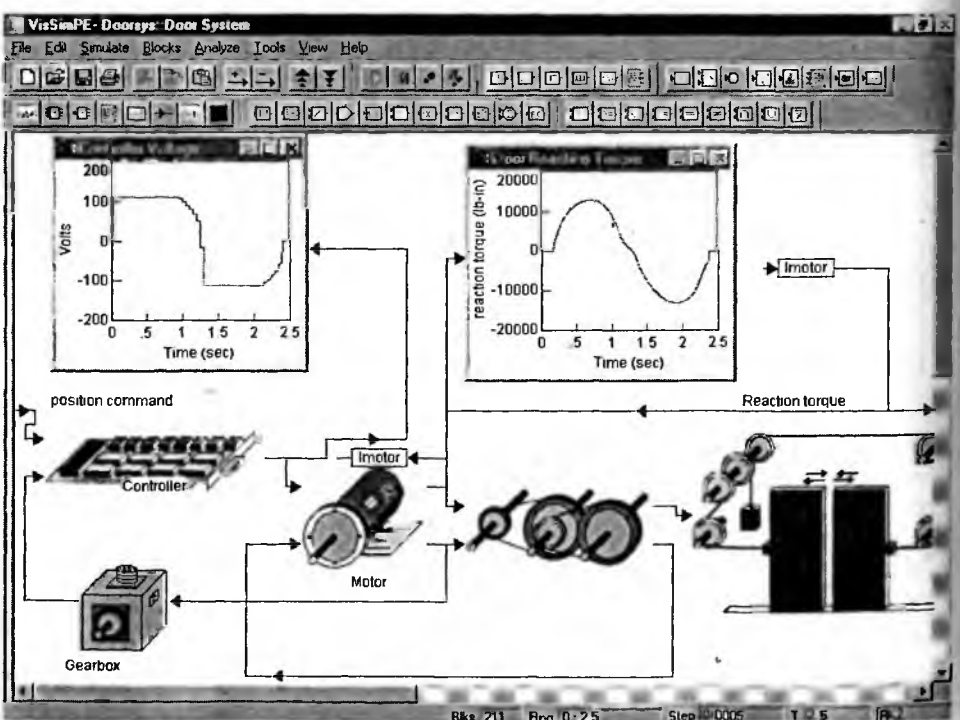


Рис. 5.85. Субблок Door System дверной системы

Наконец, на рис. 5.85 представлен субблок Door System. Этот блок содержит оригинально выполненную (с изображениями отдельных устройств) диаграмму системы. Назначение каждого блока этой модели

Как видно из приведенных примеров, VisSim может моделировать достаточно сложные объекты с высокой степенью визуализации моделей.

5.9. Моделирование экономических систем

В рыночной экономике объем предлагаемой на рынке продукции и ее цена определяются равновесием между спросом и предложением, устанавливающимся в ходе конкуренции ее производителей. Однако решение даже простейшей задачи на установление такого равновесия показывает сложность и неоднозначность процессов. Рассмотрим решение некоторых задач на моделирование экономических систем [9— 11].

5.9.1. Линейная модель спроса и предложения

Пусть некий крестьянин выращивает рожь на продажу. Он не делает запасов и весь выращенный урожай продает в текущем году. Решение о площади посевов в будущем году принимается исходя из цен текущего года. Естественно, чем выше цены в этом году, тем больше будет посеяно в будущем, и наоборот. Спрос на рожь зависит от цены в момент продажи. При росте цен спрос падает.

Необходимо описать поведение цен в различные годы и установить объемы производства зерна. Параметрами модели должны являться цена P (*price*) в текущий год за единицу товара (ржи), предложение S (*supply*) и спрос D (*demand*) в этот год. Сделаем следующие предположения.

- Рыночная цена P определяется равновесием между спросом и предложением, т. е. $S = D$ (крестьянин продает весь урожай текущего года по той цене, по которой его согласны купить покупатели).
- Предложение S и спрос D будущего года линейно зависят от цены P в текущем году. Чем больше цена, тем больше предложение и меньше спрос.

Необходимо описать поведение цены P в зависимости от ее первоначальной цены P_0 (крестьянин исходя из этой величины в первый год посеял соответствующее количество зерна). Очевидно, что такая модель может описывать установление цены для любого абстрактного товара, поведение производителей и покупателей которого отвечает выдвинутым гипотезам.

Зависимость спроса от цены можно описать линейной функцией

$$D = D_0 - KD \cdot P,$$

где D — спрос за текущий год; D_0 — спрос при нулевой цене за товар (максимально возможное потребление продукции); KD — крутизна линии спроса; P — цена товара.

Соответственно зависимость предложения от цены будет следующей:

$$S = S_0 + KS \cdot P,$$

где S — предложение за текущий год; S_0 — предложение при нулевой цене за товар (естественно, эта величина может быть только отрицательной или нулевой); KS — крутизна линии предложения; $P = \left|\frac{S_0}{KS}\right|$ — минимально допустимая цена товара, при которой его предложение будет больше нуля. Кроме того, $S = D$.

Модель для решение данной задачи представлена в задаче на рис. 5.86. Здесь переменная price — независимый аргумент функций — изменяется от 0 до 10 по линейному закону. Построены графики изменения спроса и предложения.

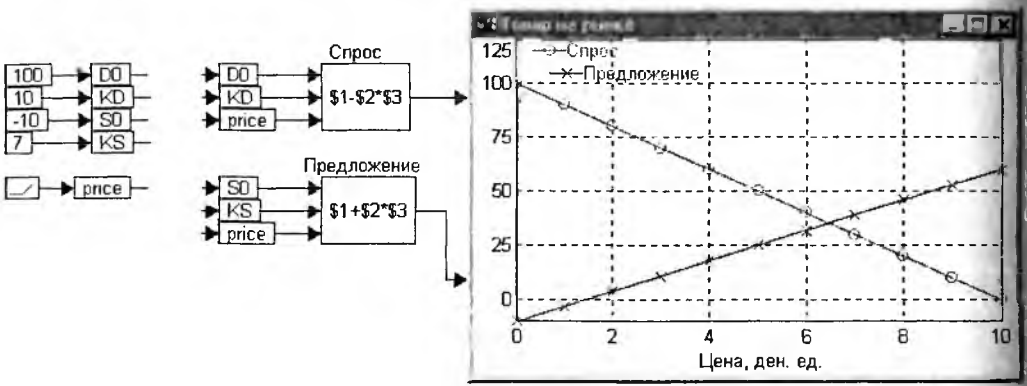


Рис. 5.86. Разомкнутая линейная модель спроса и предложения

Пересечение прямых на графике позволяет определить равновесную точку спроса и предложения, а также сделать вывод, что какой бы ни был первоначальный объем производства товара, система будет иметь устойчивое состояние равновесия — точка пересечения графиков.

Дополнив модель обратной связью (рис. 5.87), можно получить искомые зависимости цены и объема производства товара от времени.

Дискретность модели рис. 5.87 проявляется в том, что цена продажи и объем производства на будущий год определяются только один раз в год — система имеет дискретное модельное время, а в течение года параметры модели являются неизменными. Блок временной задержки e^{-sTd} предназначен в основном для анализа непрерывных систем, но, определив задержку $Td = 1$, можно обеспечить требуемую дискретность поведения модели (условно 1 год). Начальные условия для моделирования задаются также в блоке временной задержки: в окне свойств блока задается выходное значение — объем производства товара в первый год выпуска. Блоки «Спрос», «Предложение» служат для вычисления соответствующих параметров модели, блок «Вычисление цены» задает функцию, обратную функции спроса:

$$P = \frac{D_0 - D}{KD},$$

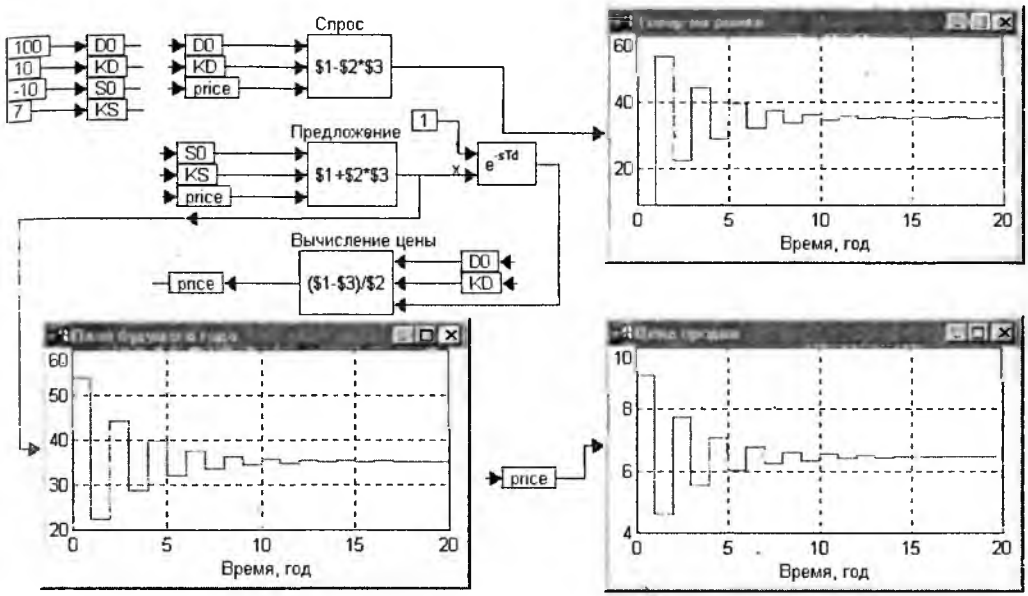


Рис. 5.87. Линейная модель спроса и предложения для случая устойчивого равновесия

с помощью которой вычисляется цена, по которой будет продан произведенный объем товара.

Оценим поведение системы в разных условиях. При $KS < KD$ система стремится к устойчивому равновесию, т. е. сбалансированному рынку, когда спрос соответствует предложению (рис. 5.87). При $KS = KD$ система находится

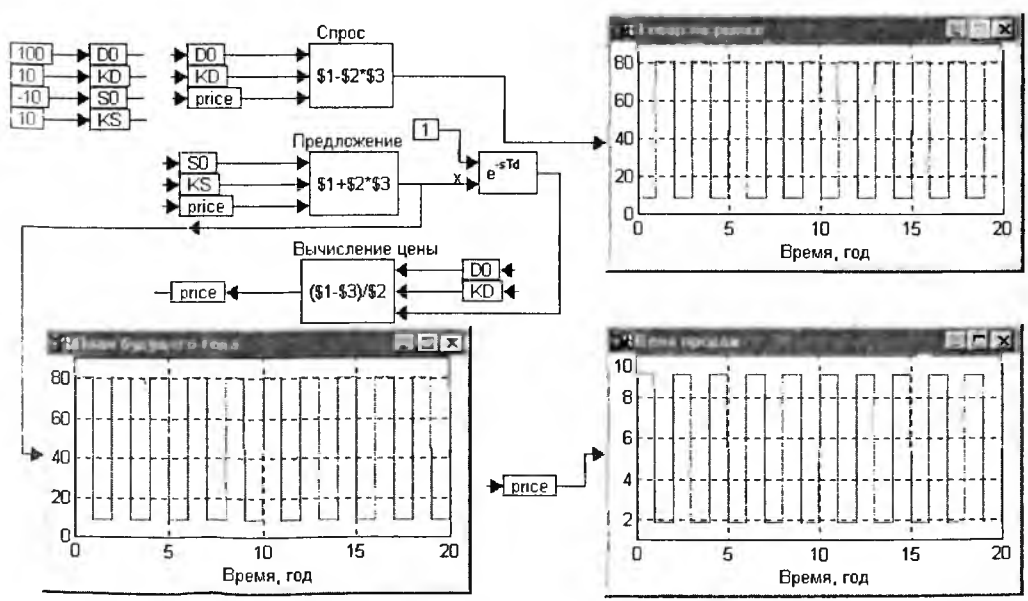


Рис. 5.88. Линейная модель спроса и предложения для случая неустойчивого равновесия ($KS = KD$)

в режиме незатухающих колебаний, идет периодическое снижение и повышение цены, а также отвечающие ей изменения объемов производства товара (рис. 5.88).

В случае $KS > KD$ — система неустойчива (рис. 5.89), рынок полностью разбалансирован, колебания нарастают по амплитуде, причем возникают отрицательные значения цены и количества товара. Это противоречит реальной экономике и объясняется неточностью модели.

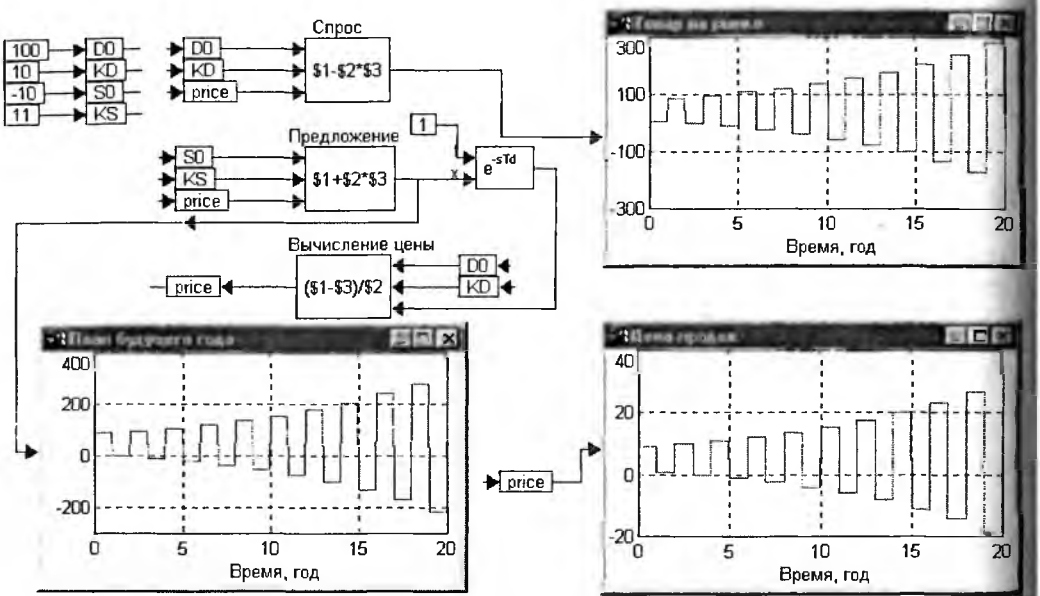


Рис. 5.89. Линейная модель спроса и предложения для разбалансированного спроса и предложения ($KS > KD$)

5.9.2. Нелинейная модель спроса и предложения

На практике очевидно, что любой производитель не может бесконечно увеличивать объем производства и функция предложения должна иметь насыщение. Спрос на товар при малых ценах растет быстрее, чем при высоких ценах, что тоже необходимо учитывать. Математически зависимости спроса и предложения можно описать выражениями:

$$D = D_0 \cdot \exp(-KD \cdot P), \qquad S = S_0 + KS \cdot P^m,$$

где $0 < m < 1$. Разомкнутая нелинейная модель спроса и предложения представлена на рис. 5.90.

Замкнутая нелинейная модель спроса и предложения представлена на рис. 5.91. Поведение системы в данном случае свидетельствует об ее устойчивости и выходе на баланс между спросом и предложением.

При моделировании нелинейной системы возможно возникновение ряда характерных ошибок, например переполнение разрядной сетки чисел и возникновение недопустимых значений аргументов у функций, задающих нелинейности, и логарифмической функции, вычисляющей цену.

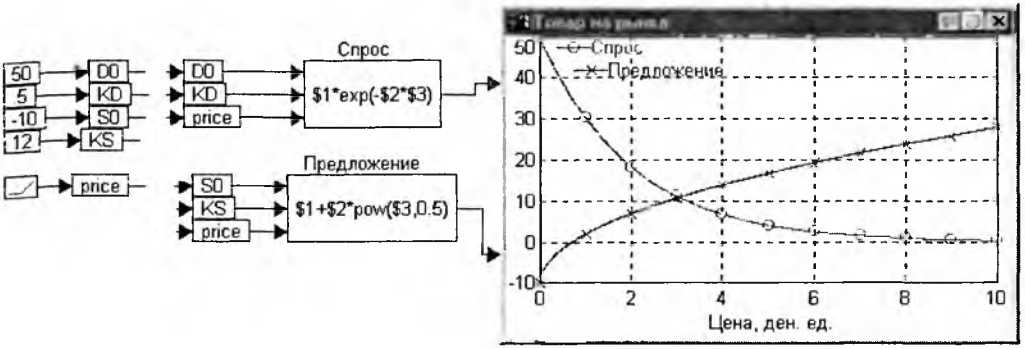


Рис. 5.90. Разомкнутая нелинейная модель спроса и предложения

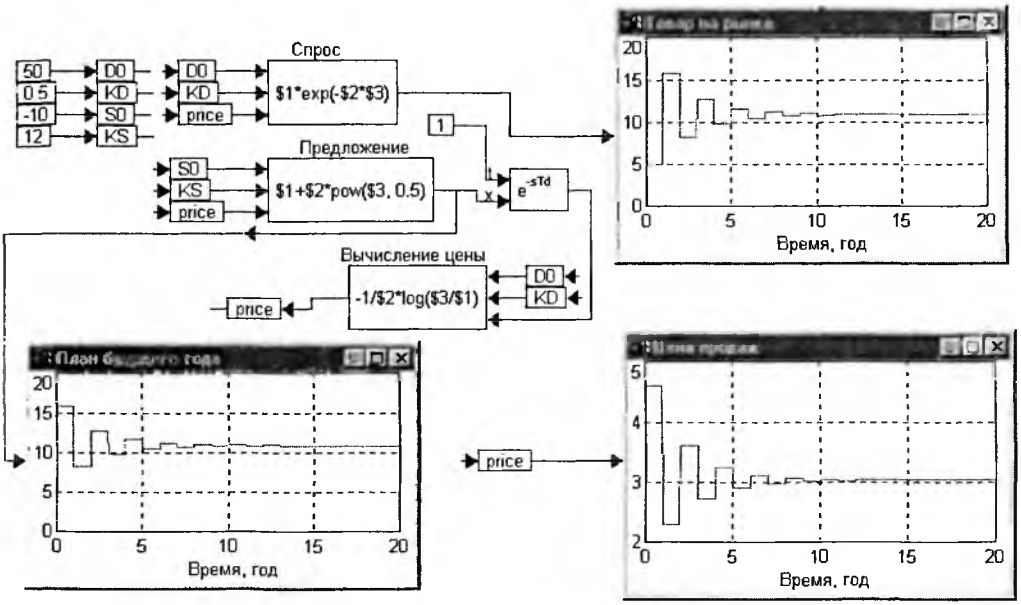


Рис. 5.91. Замкнутая нелинейная модель спроса и предложения

5.9.3. Поиск оптимальной ставки налога

Одной из важных управленческих задач в экономике является определение оптимальной ставки налогообложения прибыли. Известно, что поступления средств в бюджет будут наибольшими не при максимальной, а некоторой оптимальной ставке налога. Слишком большие налоги сдерживают развитие экономики, рост производства и вызывают уход предпринимателей в «теневой» бизнес, что и наблюдалось в нашей стране в последние годы.

Источником развития бизнеса и наполнения бюджета является превышение доходов над расходами, т. е. прибыль. Налоговые органы, или государство, устанавливают ставку налога на прибыль и получают от товаропроизводителей средства в бюджет. Каждый производитель имеет собственный капитал и из полученной прибыли отчисляет по налоговой ставке средства в бюджет.

Оставшаяся после уплаты налогов часть прибыли полностью включается в капитал предприятия. Будем считать, что прибыль предприятия пропорциональна его капиталу и рентабельности.

Прибыль предприятия за i-й год до вычета налогов можно рассчитать как

$$PR_i = CP_i \cdot RN,$$

где RN — рентабельность предприятия; CP_i — капитал предприятия в этом году.

Очевидно, что капитал в i-м году есть сумма

$$CP_i = CP_1 + \sum_{t=0}^{t=i-1} PR_t \cdot (1 - TAXE),$$

где CP_1 — начальный стартовый капитал; $TAXE$ — ставка налогообложения. Прибыль до начала работы предприятия $PR_0 = 0$ — в первый год капитал сформирован только исходными средствами.

Сумма налоговых поступлений в бюджет к i-му году включительно:

$$BD_i = \sum_{t=1}^{t=i} PR_t \cdot TAXE.$$

Решение задачи показано на рис 5.92. Расчетная модель достаточно проста. Переменные на схеме имеют те же имена, что и в математической модели. Для реализации дискретных интеграторов используются блоки цифровой задержки «1/Z». Блок «Капитал» задает значение первоначального капитала производителя. Для продвижения дискретного времени служит блок *pulseTrain*, определяющий переменную *TimeDiskret*.

Этот документ демонстрирует технику многовариантных расчетов в среде пакета. Встроенная системная переменная *$runCount* имеет значение порядко-

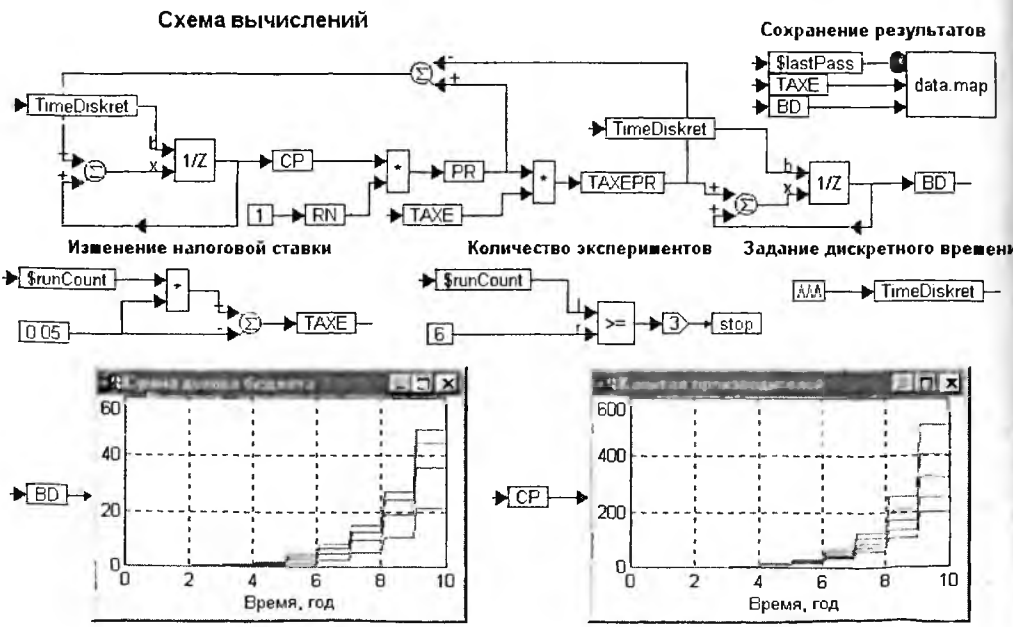


Рис. 5.92. Модель для расчета ставки налогообложения

вого номера текущего цикла моделирования, блок *stop* при входном сигнале, большем или равным 2, останавливает режим автоматического запуска циклов моделирования «AutoRestart». На выходе компаратора >= значение равно 1, при $I \geq r$ (в данном случае после 6 циклов моделирования). Эта же переменная используется для изменения значения налоговой ставки — переменной *TAXE* при каждом запуске цикла моделирования.

Служебная переменная *$lastPass* принимает значение, равное 1, в момент окончания каждого из циклов расчетов, что используется для записи конечного значения дохода бюджета *BD* вместе с соответствующей ему налоговой ставкой *TAXE* в файл **data. map**. Для сохранения результатов моделирования в файле используется блок «*export*». Графические окна блоков «*plot*» отображают изменение капитала производителей *CP* и доходов бюджета *BD* при разных значениях налоговой ставки *TAXE*.

Здесь стоит отметить недостаток пакета при проведении многовариантных расчетов, который, к сожалению, наблюдается в аналогичной ситуации у многих систем моделирования. Он заключается в том, что различные кривые на графиках никак не выделены и их можно сопоставить с изменяющимся параметром модели только по смыслу, что весьма неудобно.

Проведя с помощью модели вычисления для различных значений рентабельности производителей, сохраним зависимости суммарного дохода бюджета от величины налоговой ставки в файлах data1.map, ... data5.map. На рис. 5.93. представлен способ построения графиков по табличным данным в VisSim, используя блоки типа «*Look-Up Table*», предназначенные для считывания данных из файлов.

Видно, что существует оптимальное значение налоговой ставки, причем оно зависит от рентабельности производителя. Чем выше рентабельность, тем

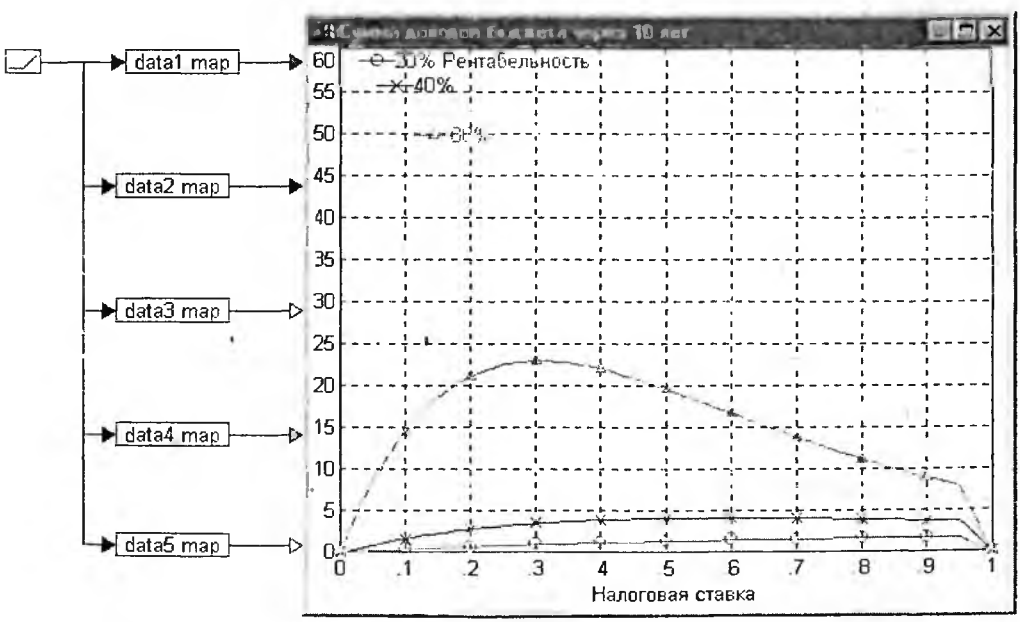


Рис. 5.93. Зависимости доходов бюджета от ставки налогообложения

меньше оптимальная ставка и более ярко выражен максимум доходов бюджета. Таким образом, ставка налогообложения должна учитывать рентабельность предприятия. Чем прибыльнее работает предприятие, тем меньше должна быть для него доля налоговых отчислений. Вместе с тем, устанавливая налогообложение, необходимо учитывать социальное и государственное значение различных производств, в противном случае в экономике останутся лишь высокодоходные алкогольная и нефтегазовая отрасли. Новые предприятия, в том числе выпускающие технологически сложную, наукоемкую продукцию, в начале проникновения на рынок принципиально не могут быть высокорентабельными.

5.9.4. Моделирование экономических кризисов

Экономисты выделяют различные виды кризисов: энергетические, структурные, кризисы перепроизводства и т. д. Увы, практически все из них можно наблюдать на протяжении двух последних десятилетий в нашей стране. Рассмотрим модель кризисов перепроизводства, которые периодически возникают в различных отраслях экономики.

Для примера возьмем ситуацию в автомобильной промышленности. Пусть есть некоторый постоянно растущий спрос $D(t)$ на автомобили. С некоторой временной задержкой Ts, требуемой на организацию и выпуск продукции, на этот спрос реагирует производство, формируя предложение:

$$S(t) = D(t - Ts).$$

Общее количество произведенных автомобилей к моменту времени t образуют величину

$$P_{in}(t) = \int_0^t D(t - Ts)dt.$$

Необходимо учесть, что с учетом срока службы Tr автомобили постоянно выбывают из общего количества выпущенных автомобилей:

$$P_{out}(t) = \int_0^{t-Tr} D(t - Ts)dt.$$

Таким образом, количество автомобилей в эксплуатации:

$$P_e = P_{in} - P_{out}.$$

Сам же спрос на автомобили есть разность между растущей потребностью $N(t)$ в автомобильном парке и количеством автомобилей в эксплуатации $P_e(t)$:

$$D(t) = N(t) - P_e(t).$$

Очевидно, что спрос не должен быть отрицательным, поэтому $D(t) \geq 0$. Пусть функция потребности в автомобильном парке будет следующей:

$$N(t) = N_0 + KN \cdot t.$$

Вместо автомобилей может быть любая продукция, спрос и предложение на которую можно описать подобным образом.

Конечно, такая модель является очень упрощенной и может использоваться только для наглядного качественного рассмотрения процессов. Построенная по этой математической модели расчетная модель показана на рис 5.94. Назначение блоков в схеме очевидно. Заметим, что здесь используются блоки непрерывных интеграторов «1/*S*». Для простоты рассмотрения функция по-

требности в автомобилях принята $N(t) = N_0 + t$, где N_0 задается начальным состоянием блока интегратора.

С увеличением задержек производства возрастает амплитуда колебаний экономической системы, появляются промежутки отсутствия спроса и прекращения производства — кризисы (рис. 5.95). Полезно самостоятельно исследовать влияние срока службы продукции, начального дефицита (разности между потребностью и начальным парком автомобилей), вида функции потребности в продукции на поведение модели.

Рассмотренные выше экономические модели имеют достаточно широкий спектр применения. Они встречаются и в области других наук, например химии и физики.

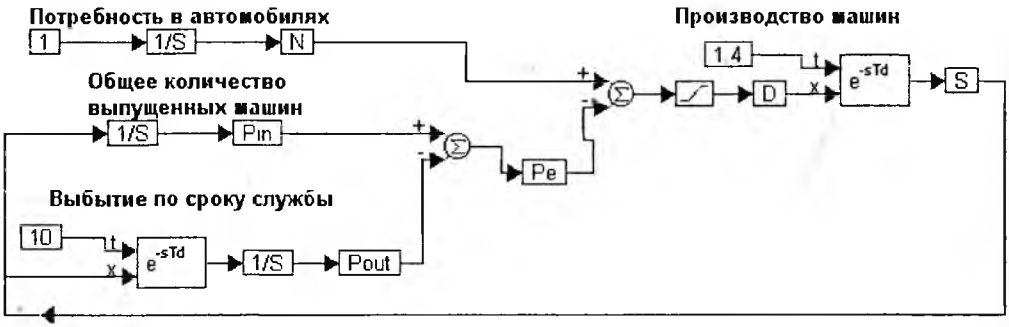


Рис. 5.94. Модель кризисов перепроизводства

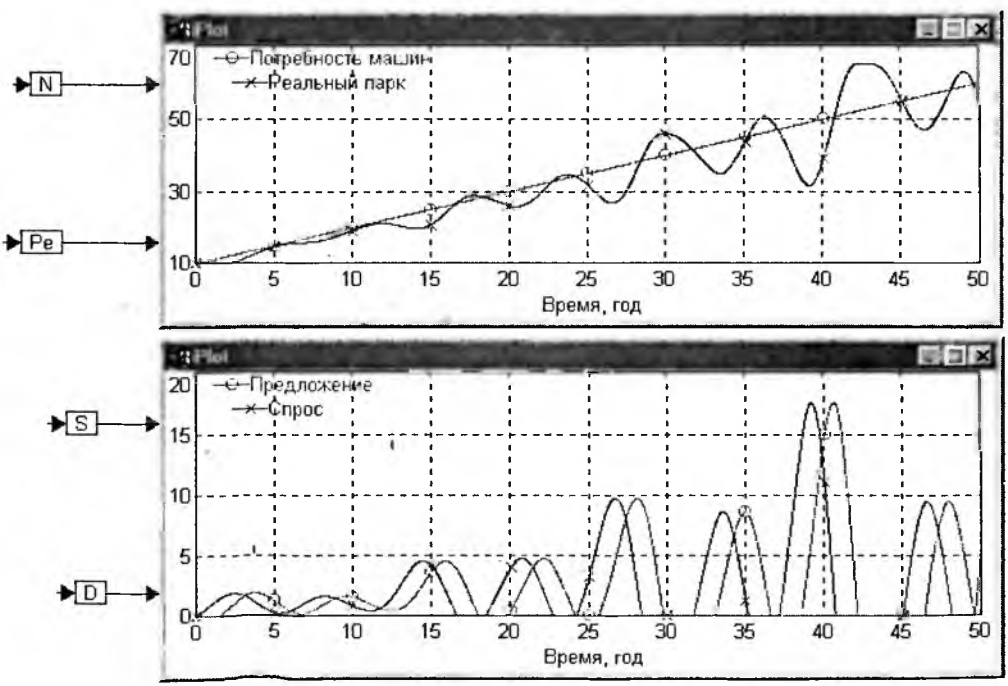


Рис. 5.95. Колебания спроса и предложения парка эксплуатируемых автомобилей

Глава 6
Интеграция VisSim с другими приложениями

Возможности системы VisSim существенно расширяются при ее интеграции с другими программными системами, в частности текстовыми и графическими редакторами и системами компьютерной математики Mathcad и MATLAB. Это позволяет создавать мощные программные комплексы по математическому моделированию, ориентированные на решение сложных задач с высокой степенью визуализации расширенным выбором методов решения таких задач. Дополнительные возможности дает применение пакета расширения VisSim/Com по проектированию и моделированию разнообразных коммуникационных устройств.

6.1. Интеграция пакета VisSim с текстовыми и графическими приложениями

6.1.1. Общие принципы интеграции

Большим достоинством системы VisSim является ее легкая *интеграция* с другими программными средствами, причем не только математического назначения, но и с популярными текстовыми и графическими программами. Под интеграцией подразумевается совместная работа интегрированных приложений с возможностью обмена данных между ними. Это позволяет повысить наглядность моделирования и использовать его при решении ряда дополнительных задач.

Основным принципом такой интеграции является установление объектной связи между системой VisSim и другим выбранным для связи приложением (OLE Object). Тем самым можно обеспечить подготовку тех или иных блоков в среде выбранного приложения и создавать гипермедиассылки на выбранное приложение. В VisSim существует два способа установления объектной связи с другими приложениями:

1) стандартный способ с помощью команды OLEobject;

2) специальный способ, обеспечивающий расширенное взаимодействие между VisSim и интегрированным с ним приложением.

Принципиальное различие между этими способами в том, что первый способ не позволяет VisSim и приложению обмениваться друг с другом сигналами, а второй способ обеспечивает возможность такого обмена.

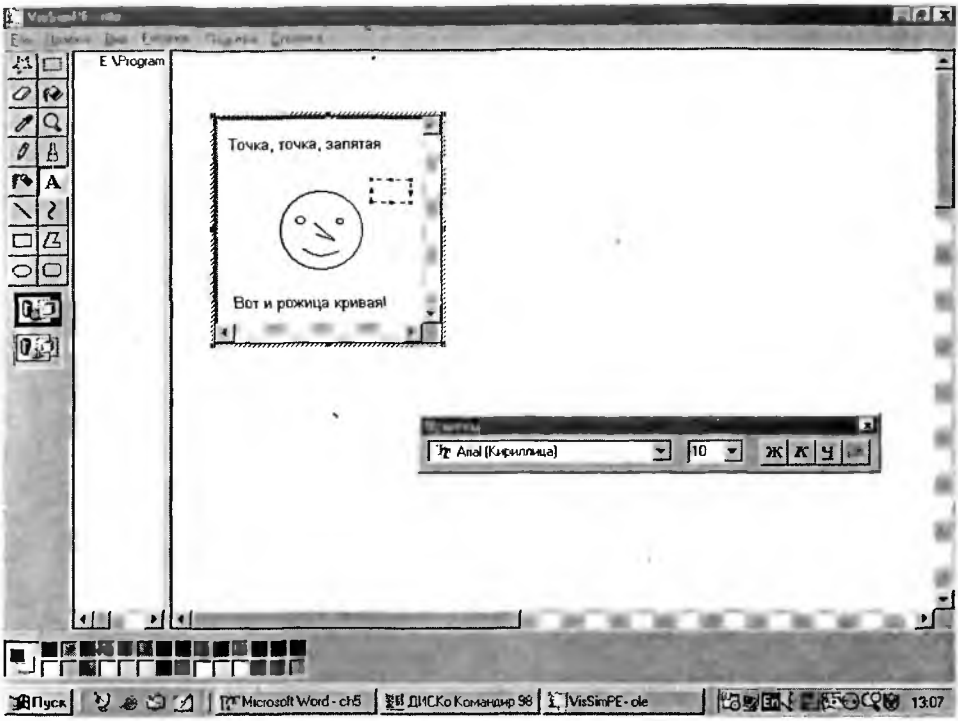


Рис. 6.2. Создание рисунка в окне редактора Microfoft Paint

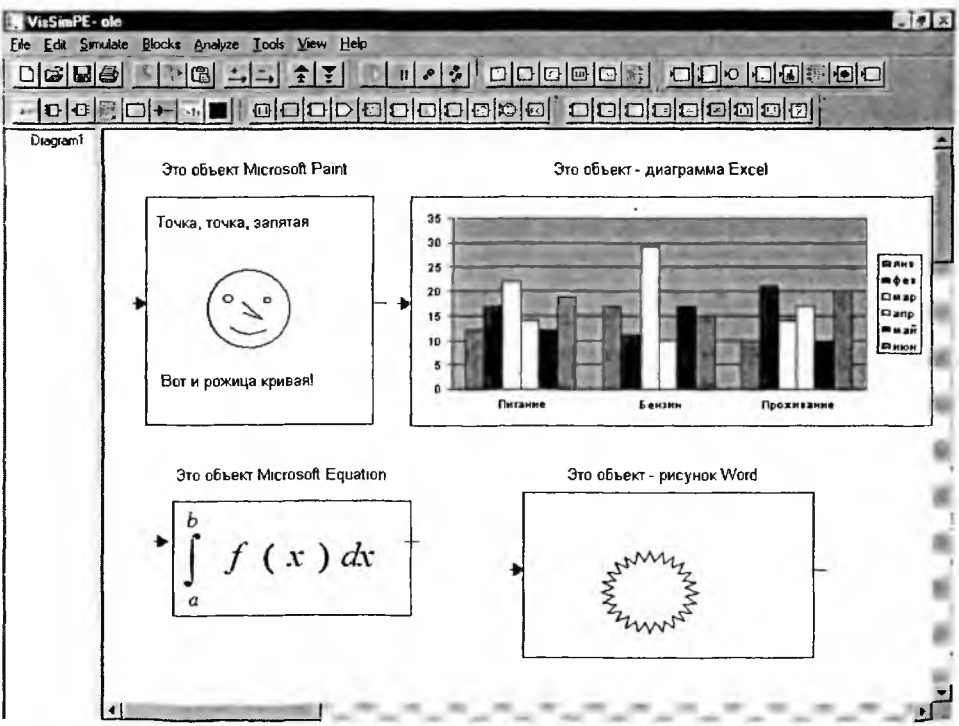


Рис. 6.3. Примеры вставок объектов различных приложений в окно модели VisSim

Для интеграции с выбранным приложением по стандартному способу достаточно создать блок этого приложения в окне модели (диаграммы) VisSim. Для этого в позиции Blocks меню имеется команда OLEobject. При ее исполнении открывается окно выбора приложения, показанное на рис. 6.1.

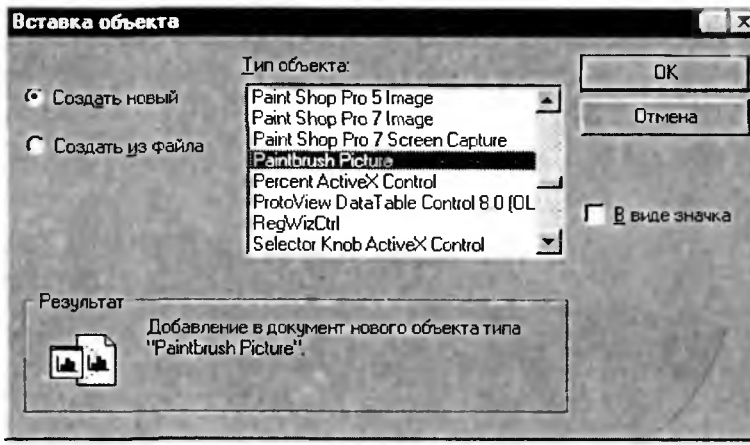


Рис. 6.1. Окно выбора приложения для объектной связи

В этом окне можно выбрать подходящее для интеграции приложение. Для этого имеется список приложений под заглавием «Тип объекта». Данное окно является стандартным окном связи между приложениями и относится к текущей версии операционной системы Windows.

6.1.2. Интеграция с графическим пакетом Microsoft Paint

Выберем, к примеру, для интеграции популярный и простой графический редактор Microsoft Paint. Тогда откроется окно этого редактора, в котором можно создавать различные рисунки, например кривую рожицу, показанную на рис. 6.2. Рисунок можно редактировать, окрашивать, снабжать надписями и т. д.

Закрыв окно приложения, с которым была установлена связь, можно наблюдать появление блока этого приложения в окне диаграммы VisSim (см. блок с нарисованной кривой рожицей в левом верхнем углу диаграммы VisSim, показанной на рис. 6.3). Аналогичным образом на этом рисунке вставлены объекты — диаграмма табличного процессора Excel, формула (интеграл), подготовленная в редакторе формул Microsoft Equations, и рисунок из набора рисунков текстового процессора Word.

Блок заданного приложения можно перемещать мышью по экрану в пределах окна модели VisSim, а также растягивать или сжимать в различных направлениях. Активизировав блок мышью, можно вернуться в окно приложения и выполнить редактирование содержимого его блока.

Любопытно отметить, что не все приложения могут быть интегрированы таким общим способом. Например, приложения матричной системы MATLAB в списке объектов (рис. 6.1) отсутствуют. Но VisSim допускает интегра-

цию с этой мощной системой, если указать это при инсталляции VisSim. При этом в позиции Blocks меню появляется соответствующее подменю.

Итак, VisSim легко интегрируется со многими программами. Это открывает обширные возможности в оформлении моделей. Особое значение, конечно, имеет интеграция с популярными приложениями, такими как текстовый процессор Word, система просмотра документов Adobe Acrobat, графическая система Visio, презентационная система PowerPoint и др. На них мы остановимся немного подробнее.

6.1.3. Интеграция VisSim с текстовым процессором Word

Все профессионально составленные и законченные модели VisSim имеют текстовые комментарии. Такие короткие комментарии удобно готовить прямо в среде VisSim. Однако длинные комментарии или комментарии повышенного стилевого качества гораздо удобнее готовить в массовом текстовом процессоре Word. На рис. 6.4 в левом верхнем углу модели показан блок связи VisSim с приложением Word.

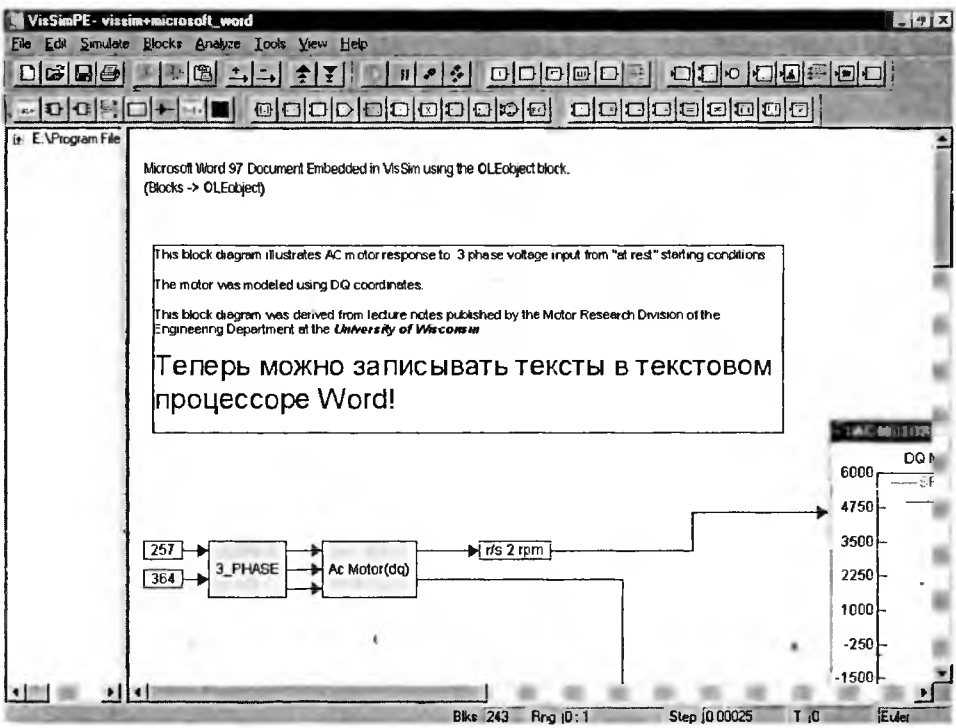


Рис. 6.4. Модель с блоком связи VisSim с текстовым процессором Word

Обратите внимание, что надписи в этом блоке сделаны на английском (остались от оригинала) и русском языках. Активизируя блок мышью, можно наблюдать переход в интегрированное приложение (в нашем случае Word), что позволяет редактировать надписи, менять их стиль, цвет и иные парамет-

ры, что в примитивном строчном текстовом редакторе VisSim делать невозможно. Вид окна VisSim в режиме редактирования надписей средствами текстового процессора Word показан на рис. 6.5.

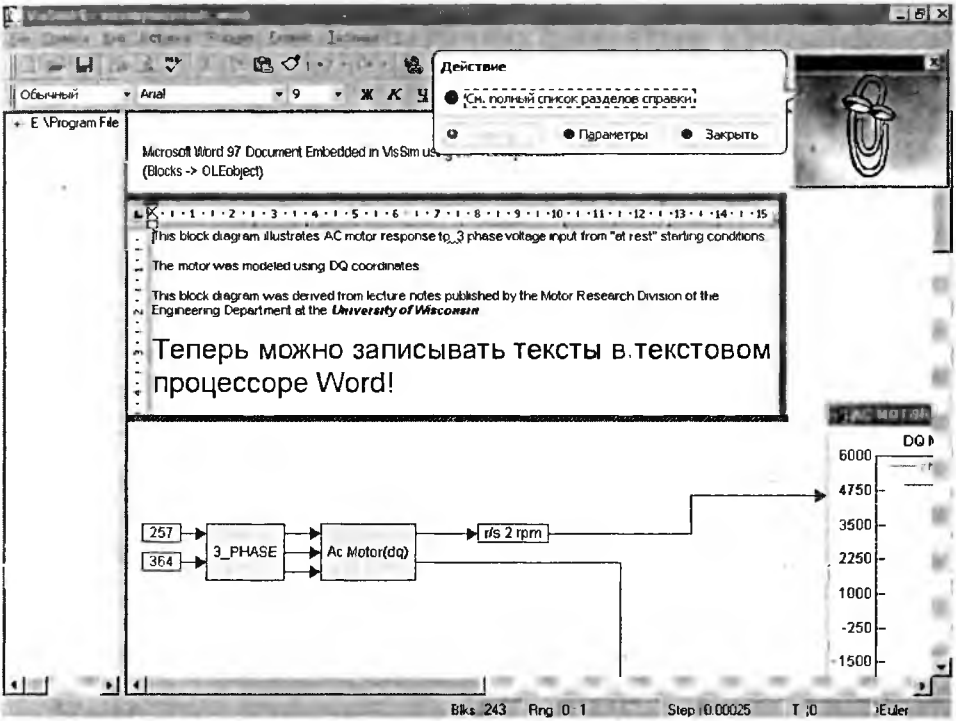


Рис. 6.5. Редактирование надписей для VisSim средствами текстового процессора Word

Описанная возможность особенно ценна при использовании VisSim в учебных целях, поскольку в этом случае детальные и аккуратные текстовые комментарии особенно необходимы.

6.1.4. Интеграция VisSim с приложением Adobe Acrobat

Довольно часто в моделях бывают полезны ссылки на те или иные документы. Многие документы хранятся в сжатых файлах специального формата PDF и просматриваются и редактируются с помощью программы Adobe Acrobat. Есть и ее версия Adobe Acrobat Reader, которая предназначена только для чтения документов. Она меньше по размерам и относится к бесплатным программам.

VisSim может интегрироваться с этой программой. Рисунок 6.6 демонстрирует вызов Adobe Acrobat прямо из диаграммы модели и просмотр с ее помощью одного из файлов по приложению Analyze.

Интеграция VisSim с Adobe Acrobat носит ограниченный характер и полезна только как средство оперативного открытия документов, созданных в формате этой программы.

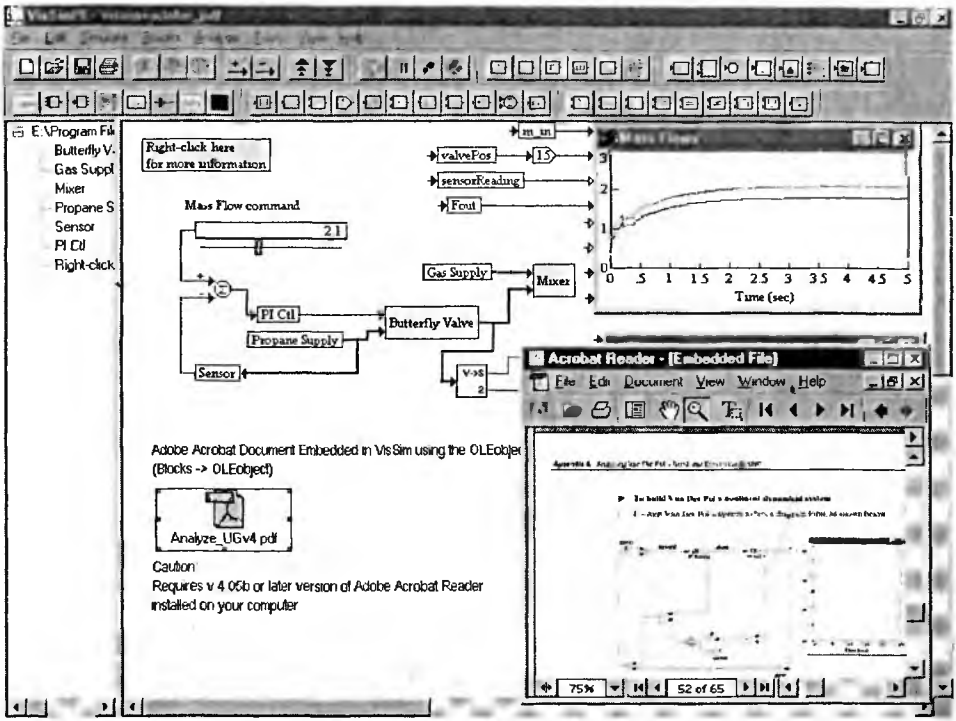


Рис. 6.6. Вызов документа формата PDF из окна модели VisSim

6.1.5. Интеграция VisSim с приложением Visio

Для профессиональной подготовки технической графики (в том числе электрических и электронных схем) широко применяется программа Visio. VisSim, кстати, как и система Mathcad, легко интегрируется с этой программой. Рисунок 6.7 показывает модель термостата, в левом верхнем углу которого имеется блок вызова программы Visio.

Активизация этого блока приводит к появлению окна программы Visio, показанного на рис. 6.8. В этом окне видно также окно редактирования текстового комментария внутри окна Visio.

6.1.6. Интеграция VisSim с приложением Power Point

Из окна модели VisSim можно вызвать целую презентацию, если воспользоваться возможностью интегрирования с приложением Microsoft Power Point. В этом приложении можно подготовить набор высококачественных слайдов, содержащих тексты, таблицы, диаграммы, формулы, рисунки и иные объекты, которые можно показывать поодиночке или автоматически.

Рисунок 6.9 показывает модель демонстрационного примера, в который вставлен объект Power Point. В данном случае это заранее подготовленная презентация по интегрированному в VisSim приложению Analyze, которое было описано в предшествующей главе.

Если активизировать мышью данный объект, то начнется показ презентации. При этом, нажимая клавиши пробела или переключения страниц, можно

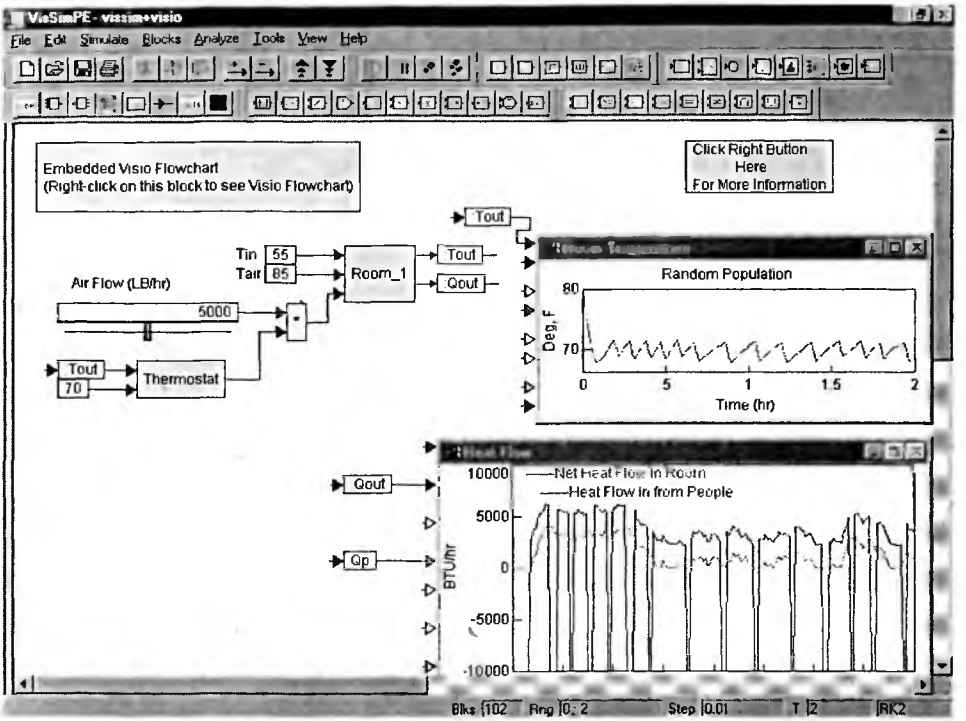


Рис. 6.7. Модель термостата с блоком вызова Visio

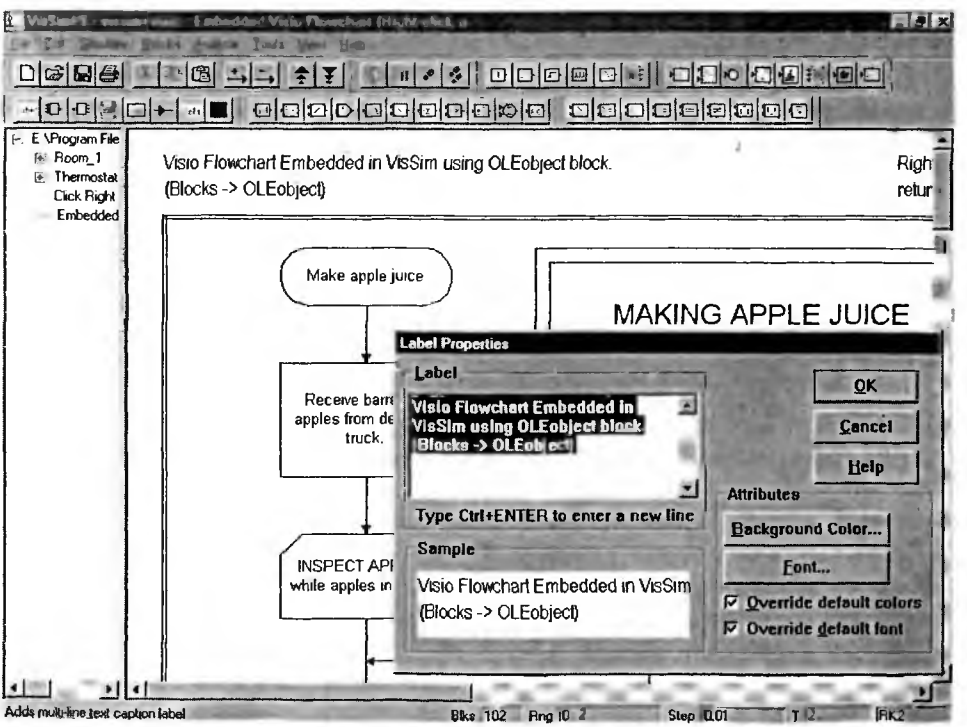


Рис. 6.8. Окно программы Visio, вызванное из окна модели VisSim.

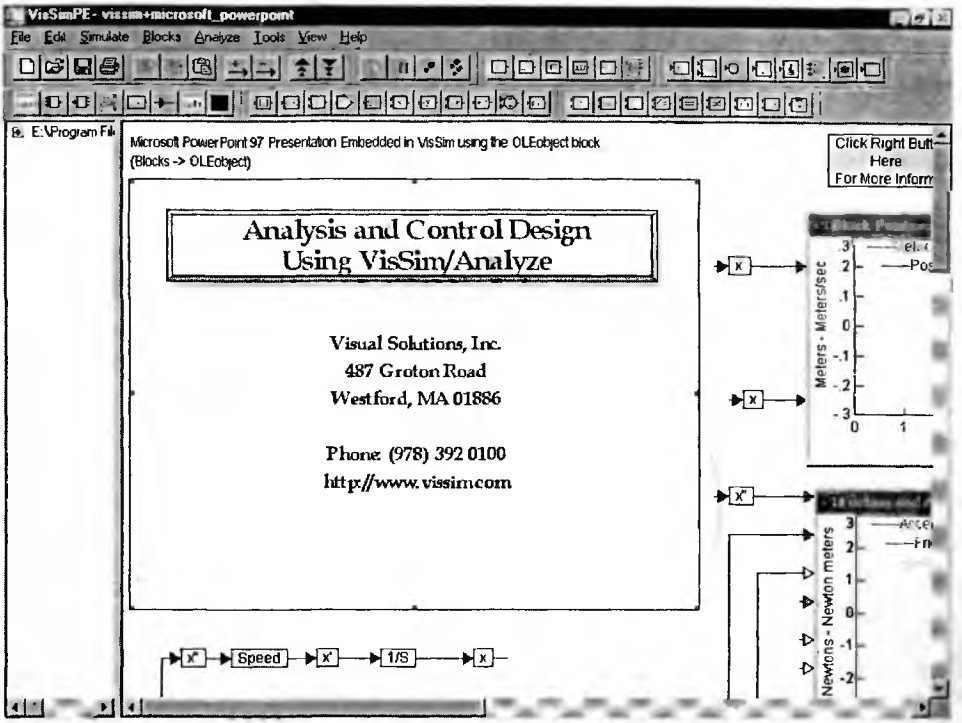


Рис. 6.9. Модель с окном презентации по приложению Analyze

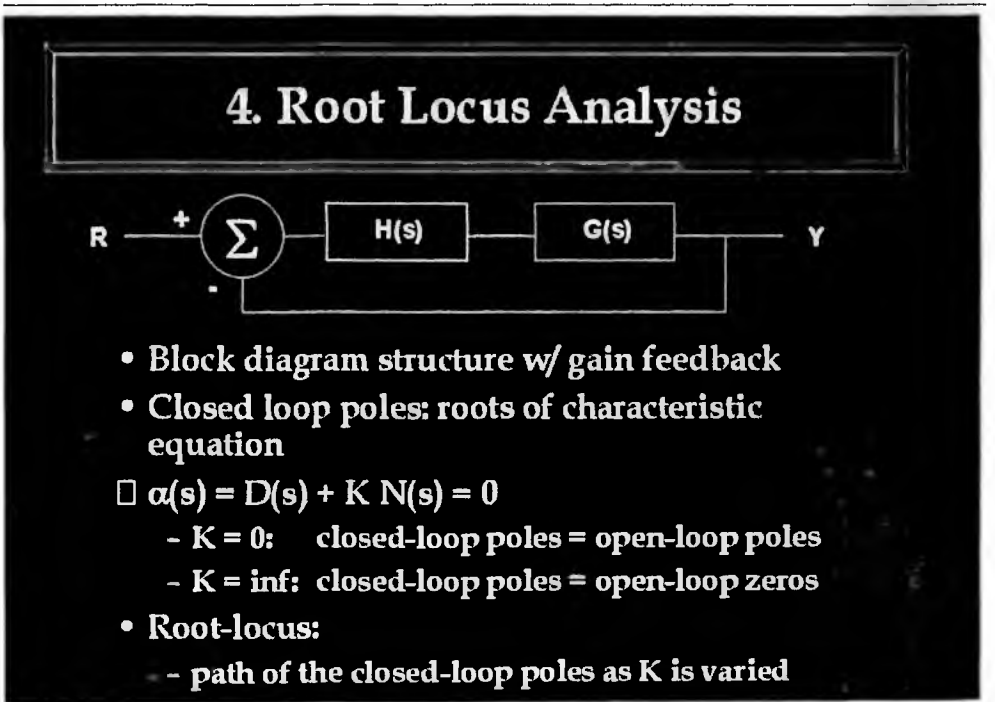


Рис. 6.10. Один из кадров презентации приложения Analyze

перемещаться от одного кадра презентации к другому и если надо просмотреть ее от начала до конца (и наоборот). На рис. 6.10 представлен один из промежуточных кадров презентации. Показ кадров в данном случае идет в полноэкранном режиме.

Для возврата из презентации в VisSim достаточно нажать клавишу Esc.

6.2. Основы интеграции VisSim с системой Mathcad

6.2.1. Характеристика и возможности системы Mathcad

Mathcad одна из самых популярных систем компьютерной математики. Эта система создана корпорацией Mathsoft (США). Число только легальных пользователей этой системой в мире составляет уже более 2 миллионов. Особенно популярна эта система в России благодаря прекрасному интерфейсу и обширным возможностям в реализации численных вычислений. Превосходна формульная и графическая визуализация вычислений, хотя форматирование графиков не очень наглядно. Уже начиная с версии Mathcad 3.0 (последняя реализация на момент подготовки данной книги Mathcad 11) Mathcad приобрел возможности символьных вычислений, реализованных на основе ядра символьных операций системы Maple (разработчик Waterloo Maple Software, Канада).

Последние версии Mathcad имеют чуть больше 300 встроенных функций. Для сравнения в системе Mathematica 4.2 их около 2000, а в Maple 8 больше 3000. Однако набор операторов и функций в Mathcad продуман настолько тщательно, что в ряде случаев важнейшие прикладные математические задачи в Mathcad решаются проще и нагляднее, чем с помощью других систем компьютерной математики. Разумеется, это придает системе Mathcad привлекательность. Можно сказать, что системы Mathematica и Maple ориентированы прежде всего на пользователей — математиков-профессионалов, тогда как Mathcad и мощная матричная система MATLAB — в первую очередь для инженеров и специалистов по прикладному применению математики.

Многие задачи математического моделирования можно решать в среде Mathcad. Однако такой путь имеет ряд серьезных недостатков. Главный из них заключается в том, что Mathcad не позволяет строить модели набором их блоков и не автоматизирует подготовку системы уравнений их состояния. Все это надо делать пользователю. Кроме того, Mathcad не позволяет проследить напрямую (в виде видимых соединений) функциональные связи между блоками. Зато это легко обеспечивает VisSim.

В то же время Mathcad обладает большими и наглядными возможностями в задании всевозможных функций и реализации различных методов решения уравнений, в том числе алгебраических и дифференциальных. Блоки Mathcad отличаются повышенной наглядностью. Некоторые версии Mathcad поставляются с системой VisSim, что делает интеграцию этих систем рациональной и легко доступной пользователям.

6.2.2. Технология подготовки VisSim моделей с блоками Mathcad

Интеграция VisSim с Mathcad осуществляется предельно просто — в диаграмму VisSim встраиваются Mathcad блоки (для этого служит подменю Insert Mathcad Object в позиции Tools меню). Это подменю имеет две команды:

- New... — создает новый Mathcad блок (объект) в виде пустого прямоугольника или значка;
- From file... — создает ссылку на Mathcad блок (объект) в виде файла.

Далее мы будем использовать первый тип Mathcad объектов — в виде явно заданного блока, в котором видны расчетные выражения. Они могут вводиться в блок после его активизации точно так, как вводятся выражения в документы Mathcad [22]. Но есть и другой путь — нужные выражения создаются в документе системы Mathcad, которая запускается параллельно с работой системы VisSim + Mathcad. Затем нужные выражения выделяются и копируются в буфер Windows командой Copy или Cut. Вернувшись в Mathcad блок диаграммы VisSim эти выражения вносятся в блок исполнением команды Paste. Практически таким образом можно в короткое время переделать документ Mathcad в диаграмму модели VisSim.

Следует учитывать, что Mathcad блок может работать и при отключенных входах. Однако отключенные выходы, как правило, недопустимы. Они приводят к появлению ошибок в Mathcad выражениях, которые помечаются красным цветом. Во избежание этого к выходу достаточно подключить виртуальный цифровой измеритель или осциллограф.

Несмотря на простоту интеграции VisSim с системой Mathcad, эта интеграция обладает принципиально новым свойством, отсутствующим в рассмотренных выше примерах интеграции VisSim с текстовыми и графическими редакторами. Это свойство заключается в возможности оперативной передачи данных из системы VisSim в систему Mathcad во время моделирования. Это достигается введением в блоки этих систем вводов и выводов. Благодаря этому можно говорить о полноценном совместном использовании этих систем.

6.2.3. Передача данных на каждом шаге моделирования

Как уже отмечалось, Mathcad блок имеет входы, обозначаемые как in0, in1, in2 и т. д., а выходы как out0, out1, out2 и т. д. В момент создания блок имеет один вход in0 и один выход out0. Но количество входов и выходов можно увеличить (до 10).

Важным вопросом при интеграции VisSim с системой Mathcad является выбор способа передачи данных к входам и выходам блока Mathcad. Простейший способ — последовательная передача данных на каждом шаге моделирования. Его осуществление демонстрирует рис. 6.11. Здесь ко входу блока Mathcad подключен блок модельного времени, а к двум выходам — каналы осциллографа.

В самом блоке Mathcad значение модельного времени присваивается переменной t, после чего на каждом шаге моделирования вычисляются функции $\sin(t)^3$ и $\sin(t)^7$. Вычисленные значения присваиваются переменным out0 и out1. Осциллограф строит их графики.

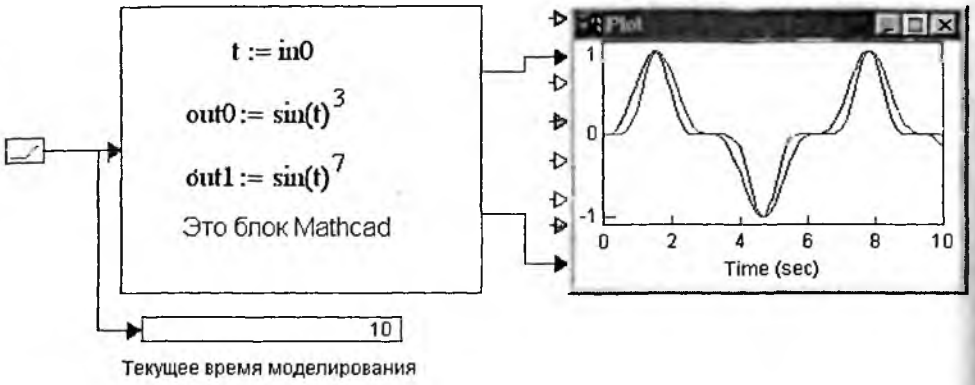


Рис. 6.11. Пример применения блока Mathcad для вычисления двух функций на каждом шаге моделирования

К сожалению, этому способу передачи данных присущ довольно серьезный недостаток — конечное и значительное время, которое затрачивается на передачу данных в блок Mathcad и вывод данных из этого блока. В итоге вычисления явно замедляются. К примеру, на ПК с процессором Pentium III с тактовой частотой 600 МГц время построения графиков при 1000 точках составило около 8 секунд. Это примерно в 10—15 раз больше, чем при решении той же задачи средствами только системы VisSim (рис. 6.12).

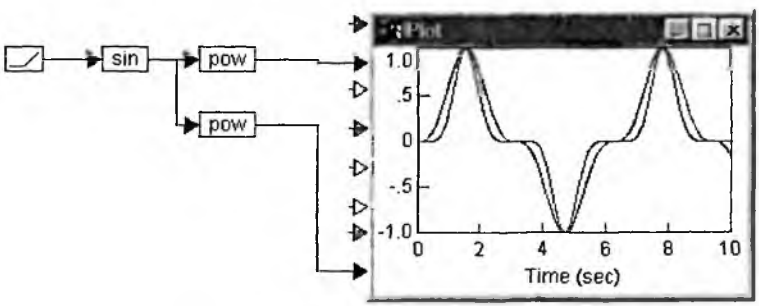


Рис. 6.12. Вычисление двух функций средствами системы VisSim

Из приведенного примера вытекает, что применять систему Mathcad в моделях VisSim стоит только в том случае, когда это действительно необходимо, например диктуется необходимостью использовать отсутствующие в VisSim средства или если наглядность вычислений стоит на первом месте и не обеспечивается при обычном построении модели из блоков.

6.2.4. Повышение наглядности вычислений

Показанный на рис. 6.13 пример на построение спирали Карно иллюстрирует повышение наглядности вычислений, достигаемое вводом Mathcad блока. В этом примере для построения спирали используются два интеграла, которые и записаны в привычном для математиков виде в окне системы Mathcad. В блок поступают данные о текущем времени моделирования, кото-

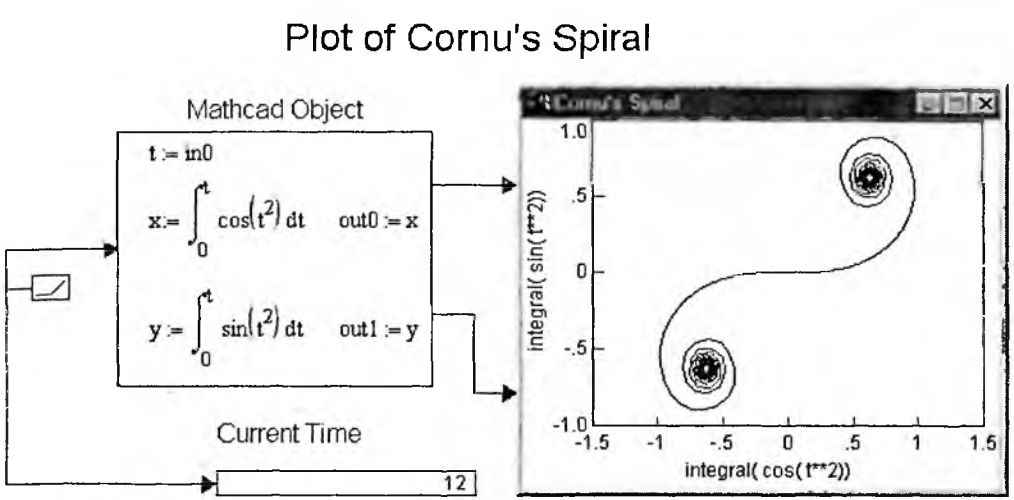


Рис. 6.13. Модель построения спирали Карно с применением Mathcad блока

рое и используется при вычислении интегралов. Значения их присваивается переменным Mathcad *x* и *y*, а затем и out0 и out1. Один из сигналов подается на вход X осциллографа, а другой на вход Y. Так что спираль Карно является фазовым портретом.

Стоит сравнить решение этой задачи с применением Mathcad блока с аналогичным решением без такого применения — средствами только VisSim. Такой пример представлен на рис. 6.14. Он, конечно, чуть менее нагляден. Но зато скорость построения спирали Карно в данном случае более чем на порядок превосходит скорость такого построения по предшествующему примеру.

Итак, представленные примеры еще раз подтверждают, что применение Mathcad в качестве средства решения задач в среде VisSim не должно быть самоцелью. Такое применение оправдано только в том случае, когда нужно воспользоваться уникальными возможностями системы Mathcad — в данном слу-

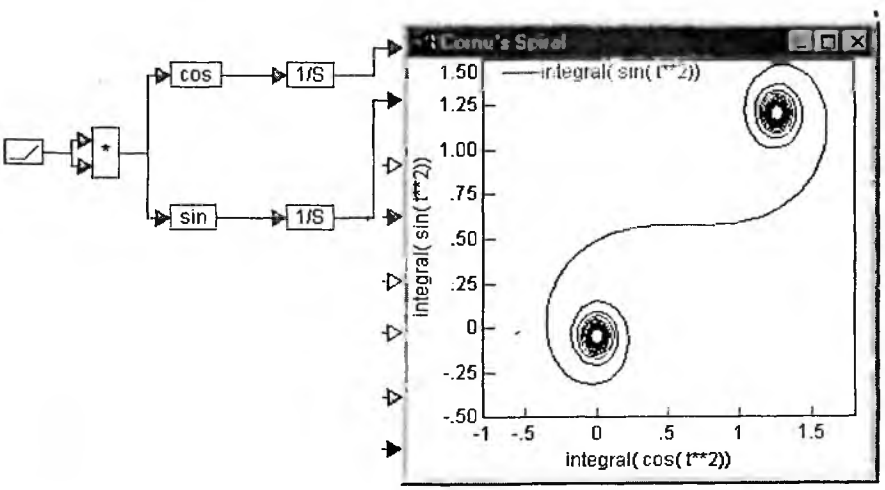


Рис. 6.14. Модель построения спирали Карно без применения Mathcad блока

чае наглядной визуализацией решения с его представлением в виде математических формул (интегралов).

6.2.5. Организация шинного вывода из блока Mathcad

Вместо вывода поодиночке данных из Mathcad блока можно организовать блоки Mathcad таким образом, что формирование массива данных будет осуществлять Mathcad, а выходы блоков будут не одиночными проводами, а шинами. При этом достаточно запустить блок Mathcad на любом, в том числе единственном, шаге моделирования. Этот прием позволяет резко (порою в десятки раз) уменьшить время вывода данных из блока Mathcad.

Пример такого решения представлен на рис. 6.15. Обратите внимание на то, что вход in0 здесь вообще не задействован. В этом примере решается задача создания двух массивов данных в среде Mathcad со значениями функции синуса и функции синуса в кубе и передача данных по двух шинам out0 и out1 на осциллограф системы VisSim.

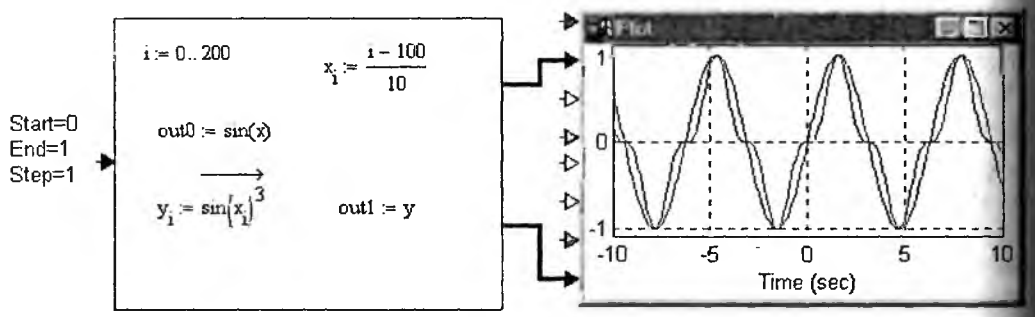


Рис. 6.15. Организация шинного выхода с блока Mathcad на осциллограф VisSim

Можно предвидеть следующую ошибку пользователей VisSim — установку параметров моделирования на обычный многошаговый режим. В этом случае вывод данных по выходным шинам будет повторяться на каждом шаге, что может привести к совершенно неприемлемому времени моделирования. Поэтому в данном примере установлены следующие временные параметры моделирования: Start = 0, End = 1 и Step = 1. При этом создается всего один шаг моделирования, но это достаточно, чтобы в блоке Mathcad было вычислено 201 значение каждой функции и передано на осциллограф для быстрого построения их графиков. Теперь оно происходит в доли секунды, т. е. практически мгновенно.

А что если задача моделирования предполагает все же пошаговое моделирование с большим числом шагов? Ответ на этот вопрос очевиден — используя логические блоки, запускайте блок Mathcad только тогда, когда это нужно, например на первом, десятом шаге и т. д. В остальное время модель может работать как обычно.

Впрочем, еще одна небольшая трудность остается — осциллограф не знает, откуда поступили данные, и даст по оси времен масштаб от 0 до 200 (тогда как в блоке Mathcad значения *x* меняются от –10 до 10). Эту трудность легко преодолеть, просто задав в окне свойств осциллографа нужный масштаб по

горизонтальной оси — от –10 до 10. Можно также изменить надписи по осям экрана осциллографа и в его пределах.

6.2.6. Шинный ввод в Mathcad блок

Иногда для увеличения объема передаваемых в блоки Mathcad данных полезна организация и шинного ввода данных в Mathcad блок. Как это делается, наглядно поясняет рис. 6.16. Тут в ходе моделирования блок буфера buffer заполняется 10 отсчетами модельного времени с единичным шагом. Сам буфер имеет емкость 16 ячеек, так что оставшиеся ячейки остаются незаполненными (обнуленными). Буфер создает вектор-строку, а операция транспонирования превращает этот массив в вектор-столбец, что и фиксирует цифровой индикатор, подключенный к выходу блока транспортирования массива.

Внутри Mathcad блока на каждом шаге моделирования формируется вектор-столбец in0, что показывает его вывод. Он передается через интерфейсную переменную out0 на выход. Этот пример хорошо проясняет работу с буфером. В частности, хорошо видно, что отсчет времени для выборок, хранящихся в буфере, идет с конца в начало. Скорость работы здесь, однако, низка, поскольку массивы данных из буфера выводятся на каждом шаге моделирования.

Рисунок 6.17 несколько расширяет представление о буферизации входа блока Mathcad. Здесь, по по данным буфера (128 ячеек, временной шаг 1), блок Mathcad вычисляет 200 значений функции синуса в кубе. Осциллографы фиксируют как заполнение буфера на каждом шаге моделирования (тут задано 200 шагов по 1 периоду модельного времени каждый), так и построение графика функции синуса в кубе.

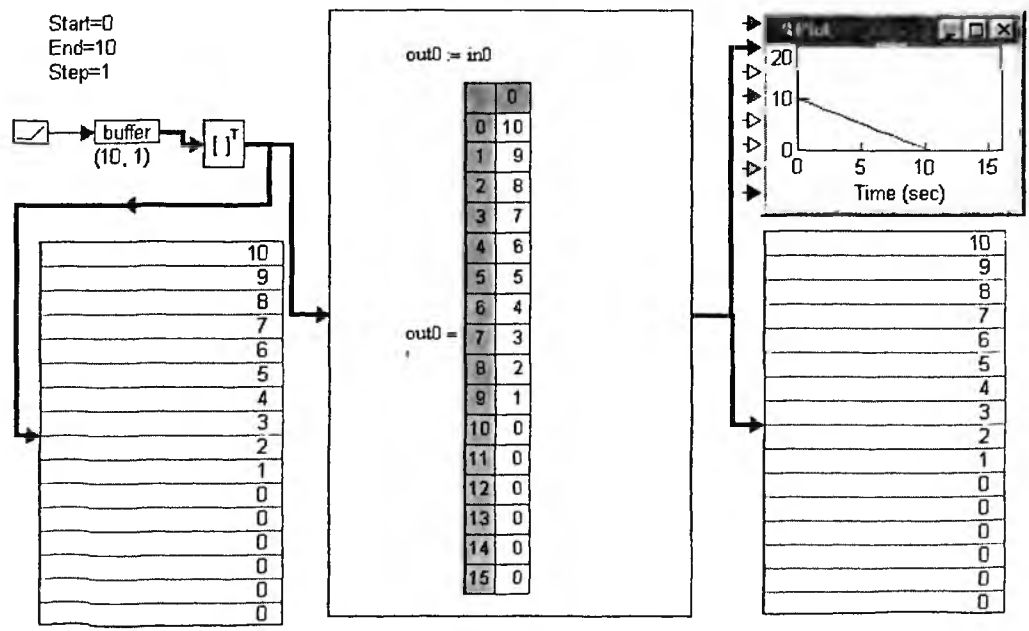


Рис. 6.16. Организация шинного ввода данных в Mathcad блок

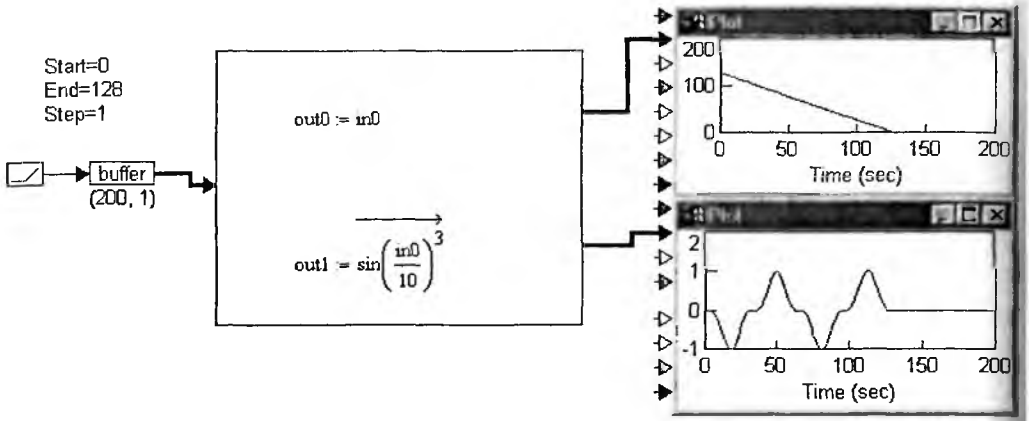


Рис. 6.17. Пример шинного ввода/вывода данных в Mathcad блок

Как и в предыдущем примере, эффективность такого ввода/вывода не высока, поскольку вычисления обновляются на каждом шаге моделирования.

6.2.7. Установка свойств Mathcad блоков

После задания Mathcad блока может понадобиться установка или изменение свойств этого блока. Для этого надо установить курсор мыши на Mathcad блок и нажать правую клавишу мыши. Если удержать ее нажатой, появится стандартное меню правой клавиши мыши, в котором есть команда Properties... . Она открывает окно свойств с двумя вкладками (на рис. 6.18 оно показано с открытой вкладкой «Общие»). Это окно информационное. Русскоязычные надписи в этом окне указывают на то, что данное окно относится к русскоязычной операционной системе Windows и используется приложением VisSim.

Рис. 6.18. Окно установки свойств Mathcad блока с открытой вкладкой «Общие»

Вторая вкладка «Вид» представлена на рис. 6.19. Здесь имеются две важные опции:

1) представить как данные — представление Mathcad блока в виде прямоугольника с данными в виде части документа Mathcad или его целиком (если документ мал);

2) в виде значка — предствление Mathcad блока в виде значка системы Mathcad или значка, выбираемого из файла, загружаемого из стандартного окна закрузки файлов, открываемого при активизации кнопки Изменить значок.

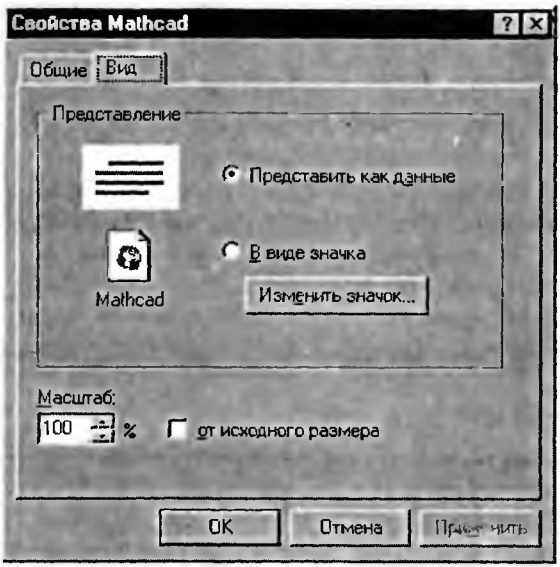


Рис. 6.19. Окно установки свойств Mathcad блока с открытой вкладкой «Вид»

Здесь также имеется список и опция для изменения размера блока, а также опция изменения размера относительно исходного размера. Действие этих опций вполне очевидно.

6.3. Техника моделирования в среде VisSim + Mathcad

6.3.1. Решение дифференциального уравнения (Rayleigh Equations)

Решение дифференциальных уравнений — одна из сфер моделирования, где система Mathcad обеспечивает повышенную степень визуализации решения и упрощает построение моделей. На рис. 6.20 показано решение специального дифференциального уравнения (Rayleigh Equations) с применением для этого Mathcad блока.

Решение здесь абсолютно прозрачно. VisSim вводит в Mathcad блок исходные параметры и выводит графики решения дифференциального уравнения. Само решение целиком осуществляется в блоке Mathcad. В нем отчетливо видно дифференциальное уравнение второго порядка и его решение с помощью блока Given, хорошо известного всем пользователям системы Mathcad.

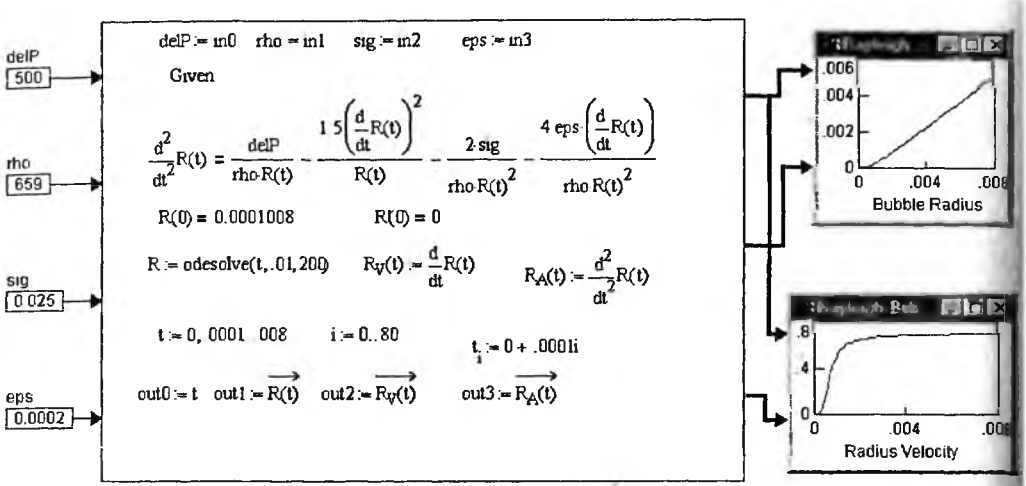


Рис. 6.20. Решение дифференциального уравнения в среде VisSim+Mathcad

Для сравнения рассмотрим решение той же задачи чисто средствами VisSim. Основная модель такого решения представлена на рис. 6.21. Она предельно проста, но в ней нет и намека на то, что и как решается.

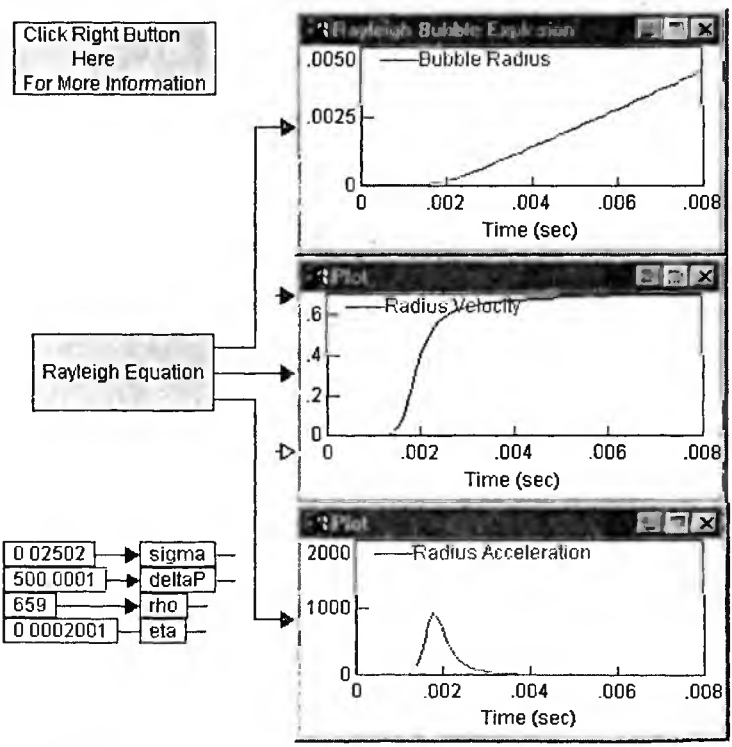


Рис. 6.21. Основная модель решения дифференциального уравнения (Rayleigh Equations) в среде VisSim

Субблок решения заданного дифференциального уравнения представлен на рис. 6.22. Естественно, что в нем также нет никаких привычных математических формул и нет в явном виде записи самого дифференциального уравнения, но зато представлена полная диаграмма реализация подготовки к численному решению этого уравнения.

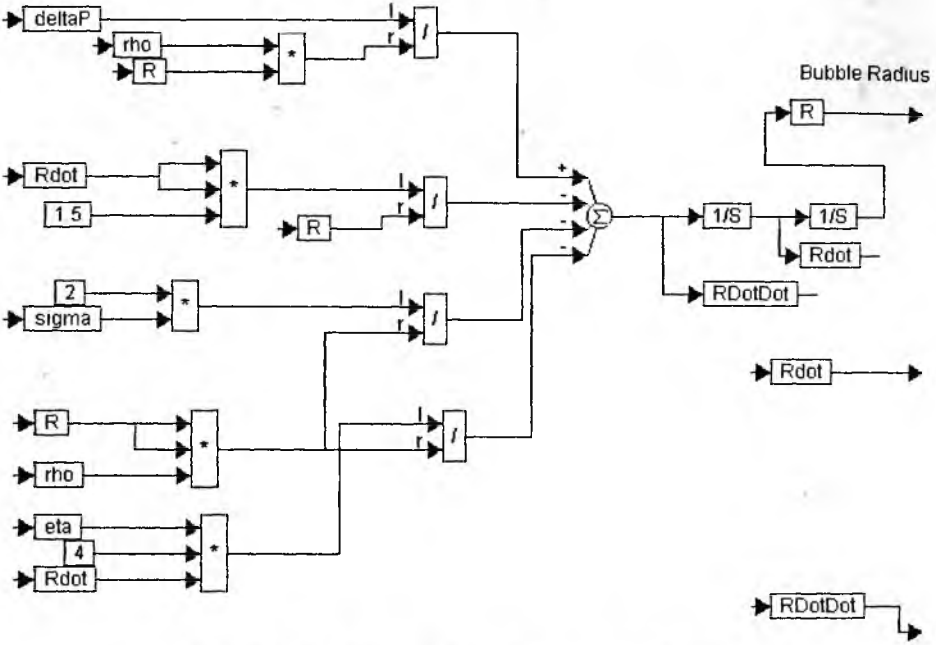


Рис. 6.22. Субблок решения дифференциального уравнения

Что здесь важнее, наглядность модели или скорость моделирования, — решать пользователю. Однако стоит еще раз отметить, что скорость вычислений во втором варианте решения заметно выше, чем в первом.

6.3.2. Моделирование кнопки с упругой мембраной

В наше время трудно представить себе более распространенное устройство, чем кнопка. Клавиатуры калькуляторов, электронных записных книжек, компьютеров и многих других окружающих нас устройств содержат десятки кнопок. Ради их удешевления применяется массовая технология изготовления кнопок в виде двух слоев резистивного материала и эластичной мембраны, которая при определенном усилии резко прогибается и замыкает слои токопроводящей пластмассы или резины. Такая конструкция кнопки позволяет получить четкое и приятное нажатие (тактильный эффект) и выполнять наборы кнопок по массовой технологии.

Рисунок 6.23 представляет простую модель такой кнопки. Плавное нажатие и отжатие во времени имитируется синусоидальной зависимостью. Сама кнопка здесь имеет реалистическое изображение, схематично показывающее устройство кнопки.

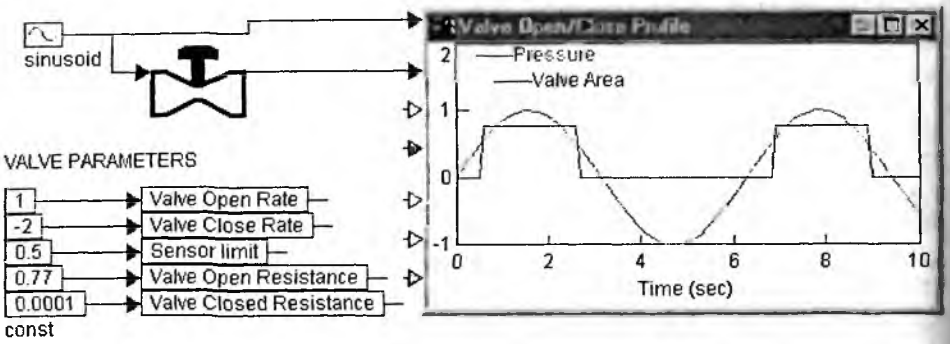


Рис. 6.23. Основная модель кнопки с упругой мембраной

Субблок кнопки, реализованный как Mathcad блок, показан на рис. 6.24. Он задает нелинейную пороговую зависимость и имитирует резкое срабатывание кнопки. Временная зависимость усилия нажатия кнопки и сигнала срабатывания отображается осциллограммами виртуального осциллографа системы VisSim.

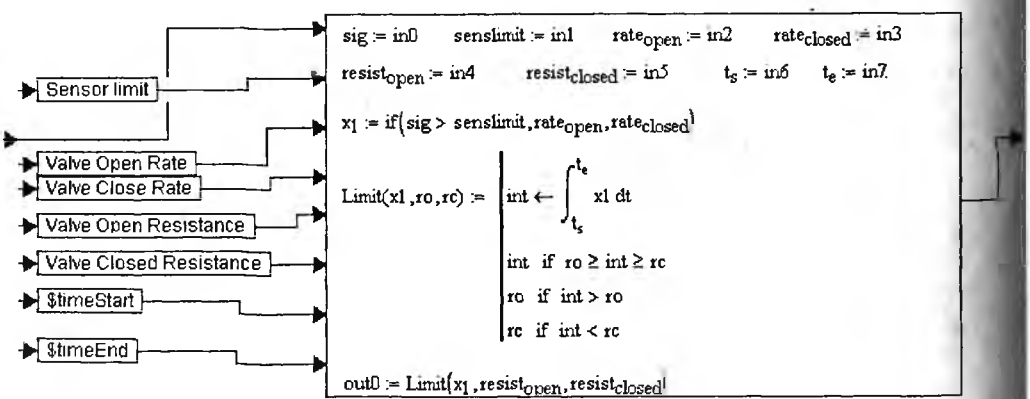


Рис. 6.24. Субблок, имитирующий нажатие кнопки, на основе Mathcad блока

Разумеется здесь используется несколько упрощенная формальная модель кнопки. Передаточная характеристика имеет релейный характер, но без учета гистерезиса. Кнопка срабатывает на включение и выключение при одном и том же пороговом усилии, да и само усилие во времени меняется по простому синусоидальному закону.

6.3.3. Моделирование кондиционера с релейным управлением

Системы управления температурой находят широкое применение. Если для малых термостатов применяются даже аналоговые системы регулирования, то кондиционирование больших помещений, как правило, основано на применении импульсных и цифровых методов.

Рисунок 3.25 демонстрирует модель релейной системы кондиционирования большого помещения, наполненного людьми (как известно, они являются источниками теплового излучения и не переносят жары). Кондиционер имеет датчик температуры релейного типа с гистерезисом, субблок которого реализован как Mathcad блок. Осциллограммы изменения температуры во времени наглядно выявляют релейный характер регулирования. При достижении уменьшающейся температурой порога в 72 градусов по Фаренгейту срабатывает релейный датчик температуры, и она начинает повышаться. Но как только приращение температуры достигнет порога в 1 градус, датчик снова срабатывает и на этот раз запускает систему охлаждения воздуха.

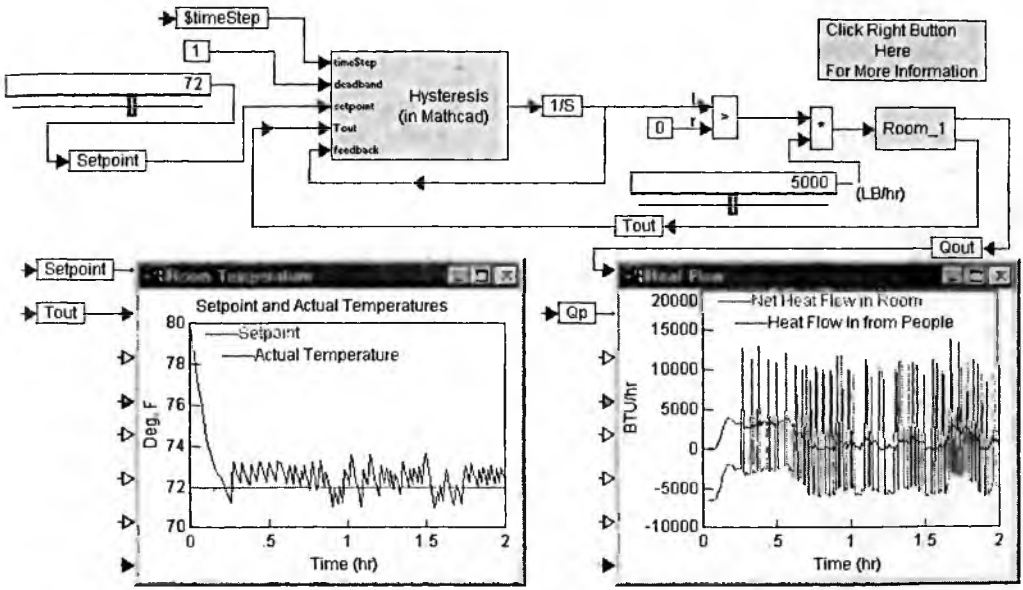


Рис. 6.25. Основная модель кондиционера

Помимо чисто информационного блока, управляемого правой клавишей мыши, эта модель имеет Mathcad субблок гистерезиса и субблок имитации помещения Room_1. Mathcad субблок (рис. 6.26) достаточно простой. Он реализует гистерезисную передаточную характеристику на основе использования простого программного модуля с функциями условных выражений.

Субблок модели помещения представлен на рис. 6.27. Собственно модель помещения представлена в верхней части диаграммы, а внизу имеется блок задания локальных параметров помещения.

Для придания моделированию большего правдоподобия в этом субблоке предусмотрен еще один субблок для задания случайно входящих в помещение субъектов. Это блок Random Population, диаграмма которого представлена на рис. 6.28.

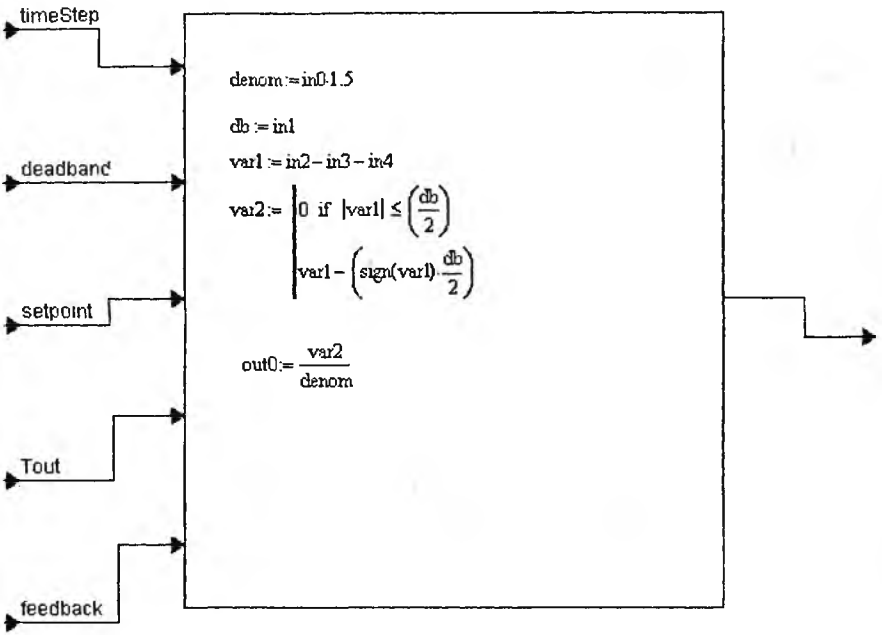


Рис. 6.26. Mathcad субблок, реализующий гистерезисную передаточную функцию

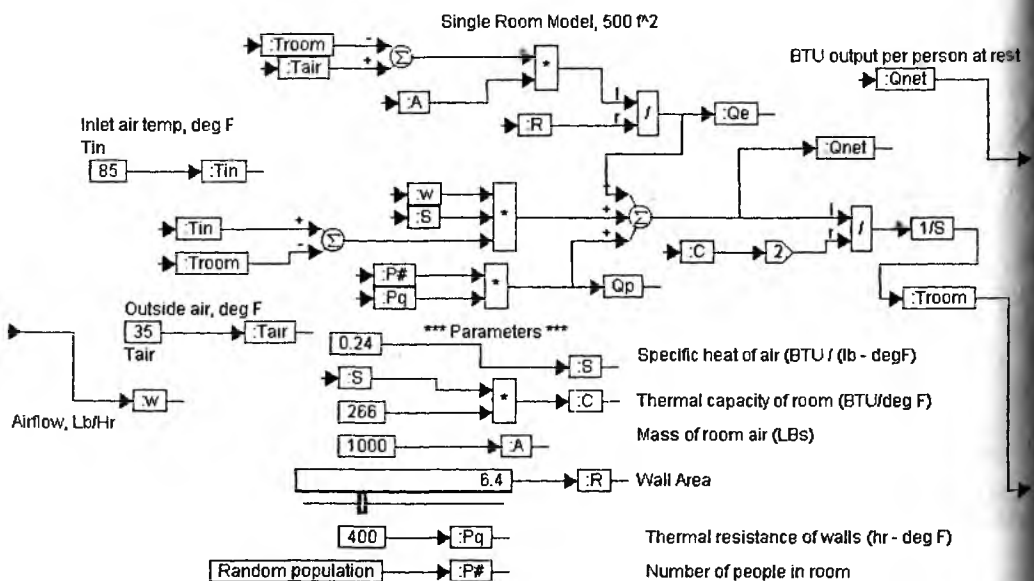


Рис. 6.27. Субблок модели одиночного помещения

Временные диаграммы работы дают наглядное представление о функционировании системы кондиционирования воздуха.

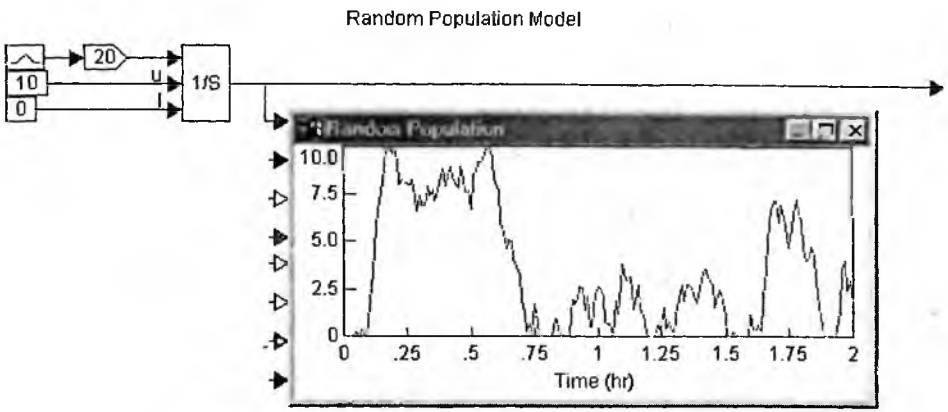


Рис. 6.28. Диаграмма субблока Random Population

6.3.4. Моделирование системы Ван дер Поля

Дифференциальное уравнение Вандер Поля — любимый тестовый пример моделирования нелинейных систем, имеющийся в примерах многих программах моделирования. На рис. 6.29 показана одна из наиболее наглядных моделей, решающая эту задачу. Она состоит из блока ввода, Mathcad блока с решением дифференциального уравнения и блока построения временных зависимостей и фазового портрета колебаний.

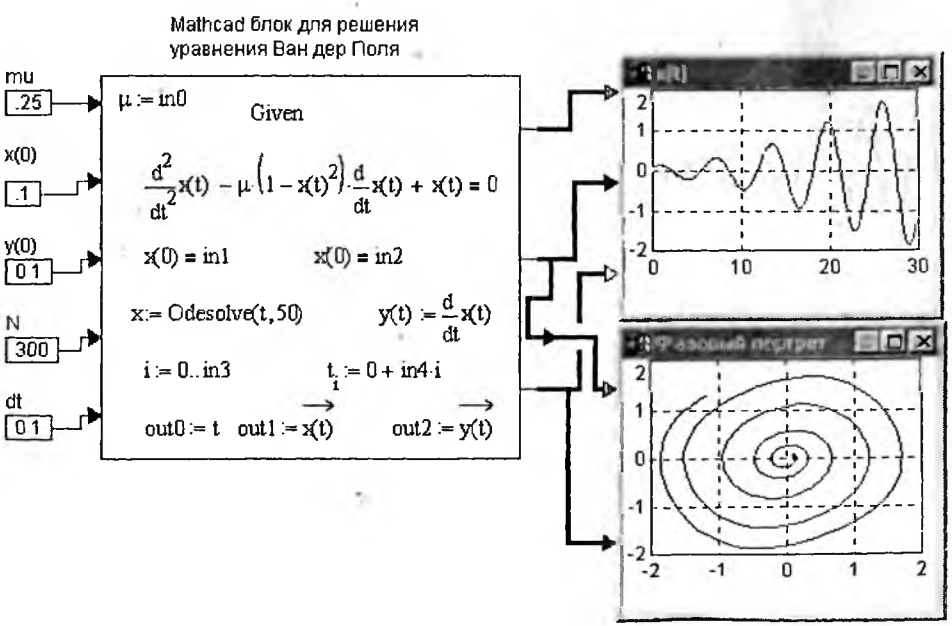


Рис. 6.29. Модель, обеспечивающая моделирование системы Ван дер Поля

Эта модель запускается при временных параметрах моделирования, равных: Start = 0, End = 1 и Step = 1 (т. е. используется единичный шаг моделирования). Модель настолько наглядна, что не требует особых пояснений.

6.3.5. Моделирование системы Лотки—Вольтерра

Поведение системы «хищники—жертвы» также описывается едва ли не в каждой книге по моделированию. Есть это описание и в книгах по системе Mathcad, например в [22]. Не повторяя его, остановимся на моделировании этой системы по уравнениям Лотки—Вольтерра, впервые описавшим возможность периодических колебаний количества хищников и их жертв. Приведенная на рис. 6.30 модель из демонстрационного примера, вошедшего в поставку системы Mathcad 2001i, моделирует данную систему, причем делает это с учетом возможности случайного изменения количества особей.

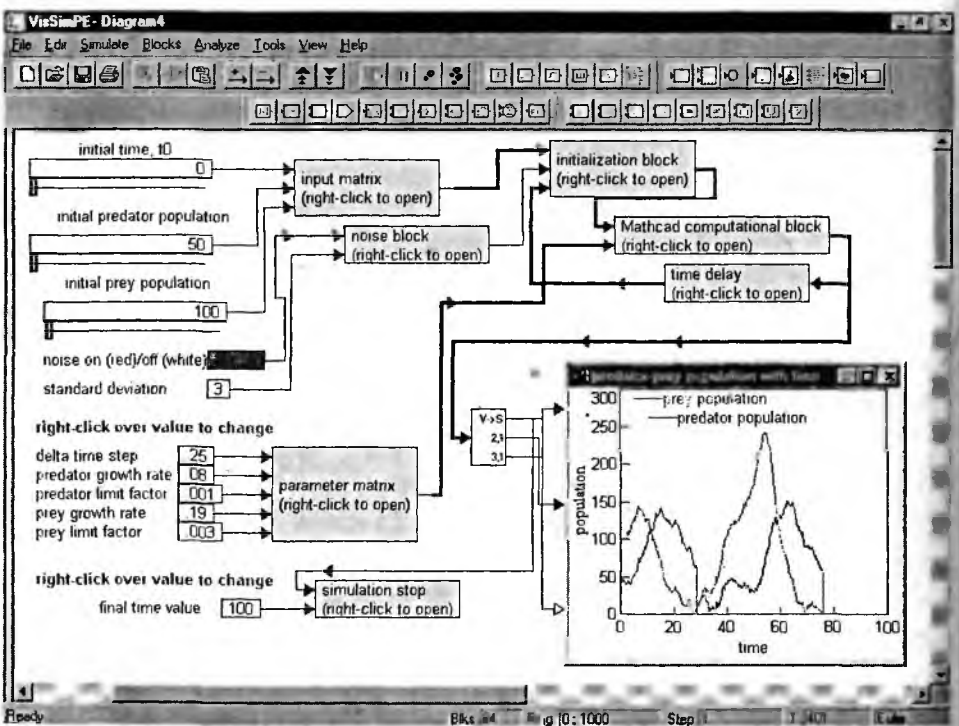


Рис. 6.30. Основной блок модели системы Лотки—Вольтерра

Не вникая в детали реализации всех субблоков этой модели, рассмотрим только Mathcad субблок, диаграмма которого приведена на рис. 6.31. Из нее можно сделать вывод, что представленная в субблоке модель системы Лотки—Вольтерра решается конечно-разностным методом, особенности реализации которого вполне очевидны.

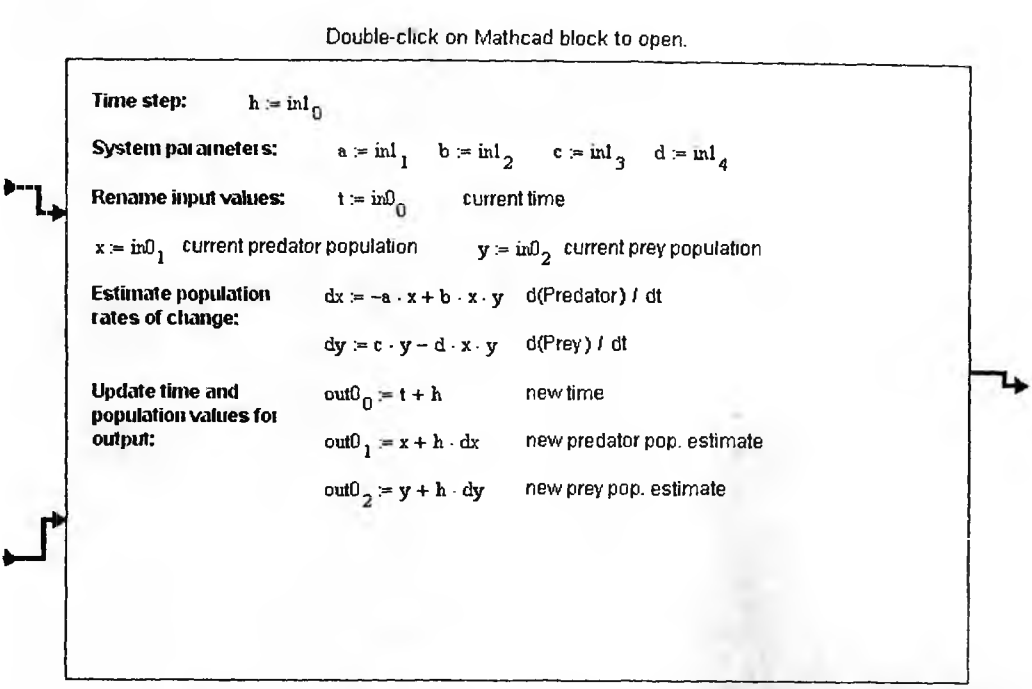


Рис. 6.31. Mathcad субблок, реализующий решение системы уравнений Лотки—Вольтерра

Остается добавить, что моделирование по данной модели идет довольно медленно, во всяком случае по сравнению с реализацией, использующей только систему Mathcad. Заинтересованный пользователь может легко реализовать подобную модель и только в среде VisSim.

6.3.6. Моделирование системы Даффинга

Примером неавтономной системы второго порядка является система Даффинга. Модель этой системы, показанная на рис. 6.32, описывается дифференциальным уравнением второго порядка, в правой части которого записано выражение $A \cdot \cos(\omega \cdot t)$. Оно задает гармоническое внешнее воздействие на систему. Его взаимодействие с собственными колебаниями системы порой создает колебания очень необычного вида (см. временную зависимость колебаний и их фазовый портрет, отображаемые виртуальными осциллографами системы VisSim).

Система Даффинга неплохо описывает явления, характерные для оптических резонаторов, используемых в лазерных установках. В частности, она демонстрирует появление паразитных видов колебаний (мод), порою напоминающих хаотические колебания.

Внимательный читатель уже понял, что в который уже раз мы используем подходящий документ системы Mathcad и просто дополняем его блоком ввода и регистрирующими осциллографами. Этот путь позволяет решать

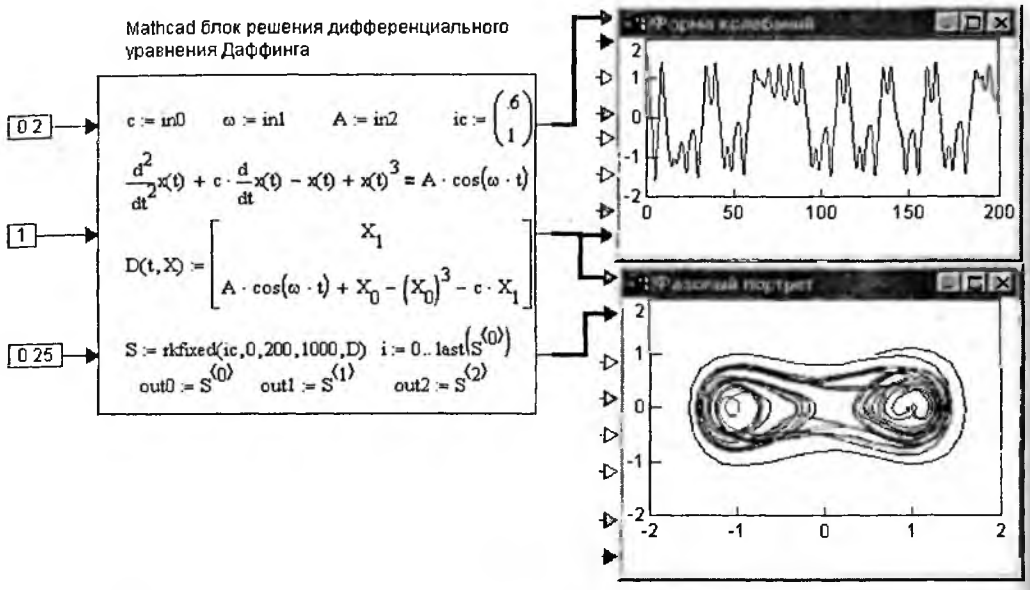


Рис. 6.32. Моделирование системы Даффинга

множество задач, на что уже было затрачено немало времени и интеллектуальных ресурсов.

6.3.7. Решение задач линейного программирования

В экономике существует множество задач в области линейного программирования. Это задачи на поиск максимума стоимости производимой на предприятии продукции, транспортные задачи и т. д. Такие задачи сводятся к решению систем линейных относительно переменных уравнений при заданных (в виде неравенств и равенств) условиях. Решение таких задач в среде VisSim довольно сложно, поэтому вполне разумно привлечь для этого систему Mathcad.

Рисунок 6.33 иллюстрирует решение типовой задачи линейного программирования. В блоке текстового комментария эта задача подробно описана. Под ним дана модель решения этой задачи. Она состоит из группы блоков ввода, Mathcad блока с решением задачи и цифрового измерителя для вывода результатов.

Специфика задач линейного программирования заключается в большом числе исходных параметров. Так, в представленном примере их 13, тогда как число входов к Mathcad блокам ограничено значением 10. Рисунок 6.33 поясняет, как легко выйти их этой ситуации; для этого надо некоторые однотипные данные (например, количества и стоимости материалов или изделий разного типа) объединить в шины. Это не только уменьшает число нужных входов, но и делает вход более понятным. Рекомендуется у входных соединений и шин проставлять комментарии о том, к чему относятся эти соединения и входы. Вывод данных в таких задачах обычно осуществляется с помощью цифрового регистратора данных.

Пусть цех малого предприятия должен изготовить 100 изделий трех типов. Каждого изделия нужно сделать не менее 20 штук. На изделия уходят соответственно 4, 3.4 и 2 кг металла при его общем запасе 340 кг, а также по 4.75, 11 и 2 кг пластмассы при ее общем запасе 700 кг. Сколько изделий каждого типа x1, x2 и x3 надо выпустить для получения максимального объема выпуска в денежном выражении, если цена изделий составляет по калькуляции 4, 3 и 2 рубля? Итак, задача сводится к вычислению максимума функции f(x1.x2,x3)=4*x1 + 3*x2 + 2*x3. Пример решения дан ниже.

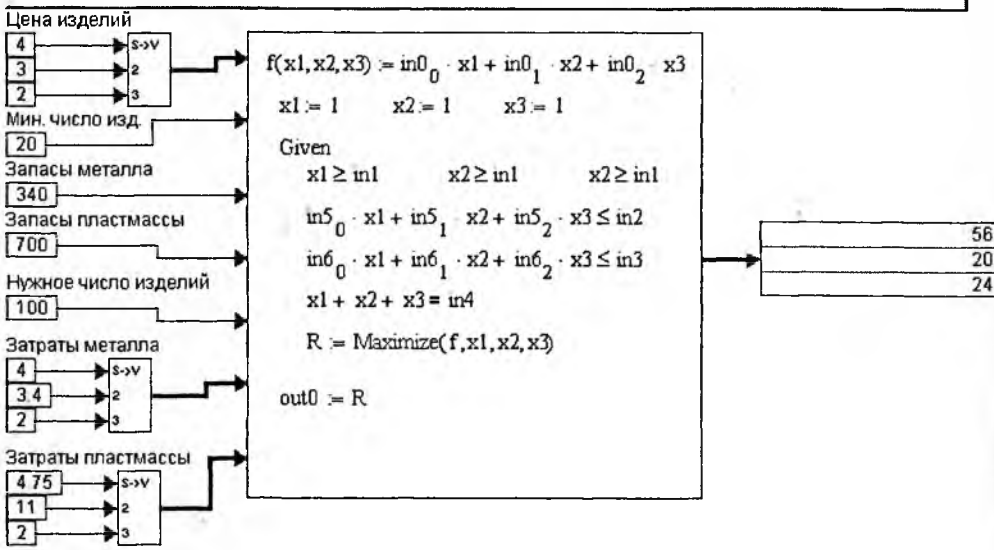


Рис. 6.33. Пример решения задачи линейного программирования в среде VisSim + Mathcad

6.3.8. Спектральный анализ и синтез с применением функций БПФ Mathcad

Mathcad (как и VisSim) имеет свои функции быстрого преобразования Фурье (БПФ). Рисунок 6.34 демонстрирует их применение в составе Mathcad блока. Сигнал задается в виде прямоугольного импульса блоками VisSim, а регистрация сигнала после прямого и обратного преобразований Фурье осуществляется осциллографом VisSim. Другой осциллограф служит для построения спектра сигнала — зависимости модуля гармоник от их порядкового номера.

Стоит отметить несколько особенностей реализации этой модели. Прежде всего надо отметить, что спектральному анализу подлежит только та временная зависимость (сигнал), которая создается на выходе буфера. Поэтому надо правильно выставлять параметры перепадов и буфера. На рис. 6.34 они даны в скобках.

В Mathcad блоке предусмотрено ограничение числа гармоник — параметр k, — которые используются для обратного преобразования Фурье. Ограничение числа гармоник вызывает так называемый эффект Гиббса — характерные колебания синтезированной по гармоникам функции. Эти колебания отчетливо видны на осциллограмме подвергнутого прямому, а затем и обратному преобразованиям Фурье импульса при числе гармоник k = 10.

Этот эффект виден даже при числе гармоник k = 28, близком к максимальному — 32 (рис. 6.35). Однако он исчезает, если прямое и обратное преоб-

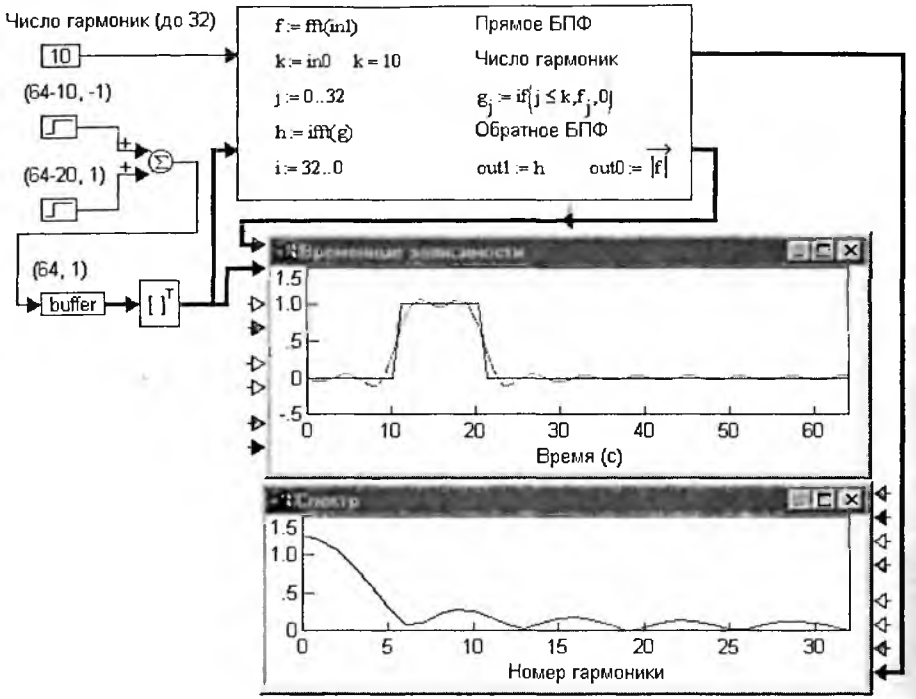


Рис. 6.34. Спектральный анализ и синтез с применением функций БПФ Mathcad при k = 10

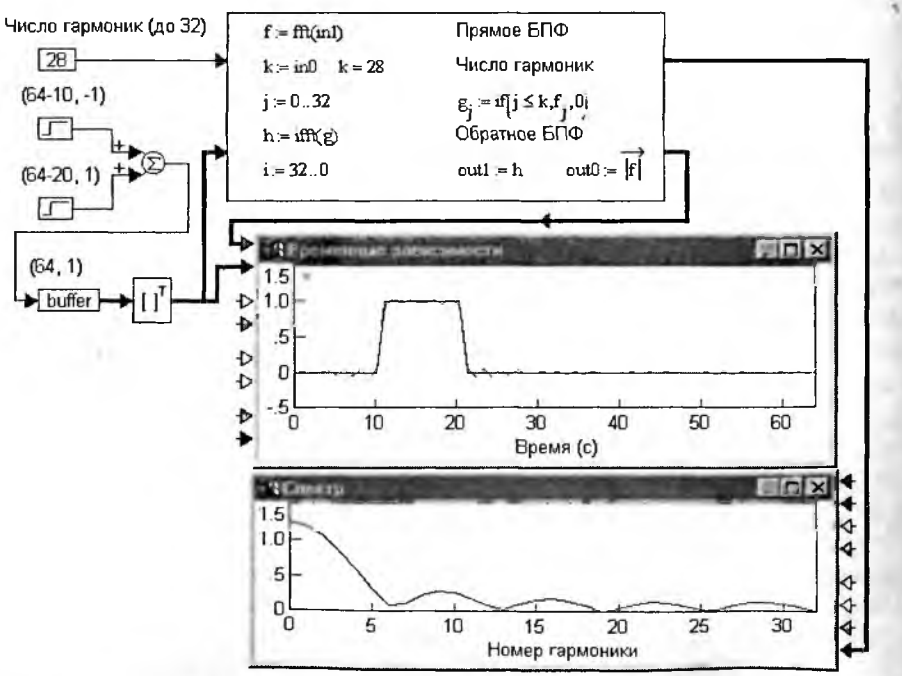


Рис. 6.35. Спектральный анализ и синтез с применением функций БПФ Mathcad при k = 28

разования Фурье выполняются без ограничения числа гармоник (т. е. при k = 32 в нашем случае). Максимальное число гармоник определяется половиной числа выборок сигнала (64 в нашем случае).

Спектральный анализ с применением системы Mathcad выполняется заметно проще и нагляднее, чем средствами VisSim. Но по скорости выполнения он все же проигрывает решению той же задачи на основе применения только системы VisSim.

6.3.9. Модель анализатора спектра сложных сигналов

В демонстрационных примерах совместного применения VisSim и Mathcad описана модель анализатора спектра сложных сигналов, применяемых для анализа химических процессор (разумеется, область применения такой модели не ограничена этим конкретным применением). Рисунок 6.36 показывает окно основной модели анализатора спектра при анализе тестового шумового сигнала, создаваемого блоками в левом верхнем углу диаграммы модели.

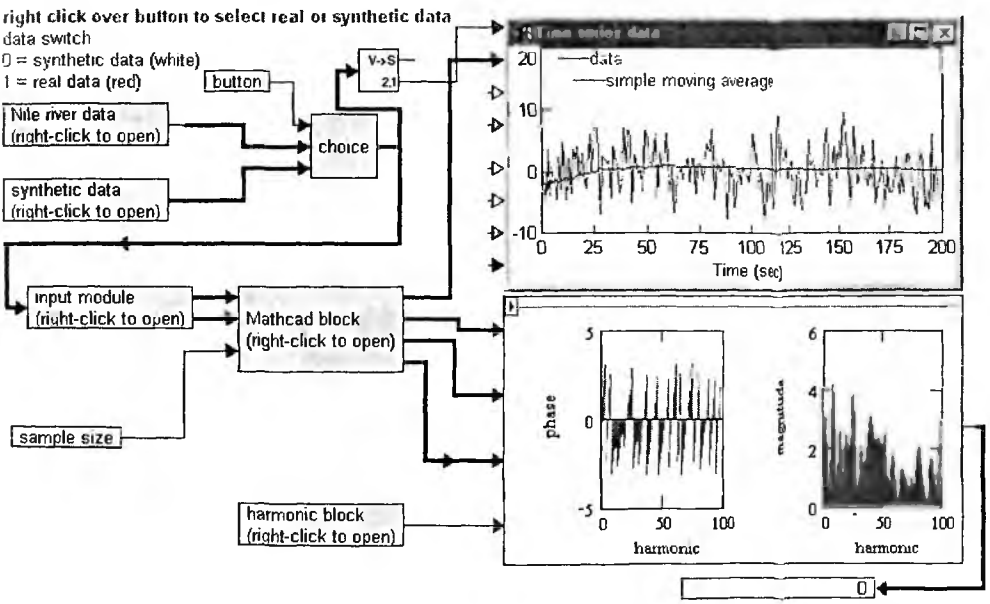


Рис. 6.36. Основная модель анализатора спектра сложных сигналов (анализ тестового сигнала)

Для переключения источников сигнала служит кнопка button. Если она отжата и имеет светлый фон, то анализируется спектр шумового текстового сигнала. Если кнопка нажата (щелчком правой клавиши мыши) и ее фон становится красным, то анализируется спектр реального сигнала, хранимого в файле данных. Этот случай показан на рис. 6.37.

Наибольший интерес в этой модели представляет субблок анализа спектров с именем Mathcad block. Если раскрыть его (щелчком правой клавиши мыши), то можно увидеть, что этот субблок состоит из двух Mathcad блоков (рис. 6.38).

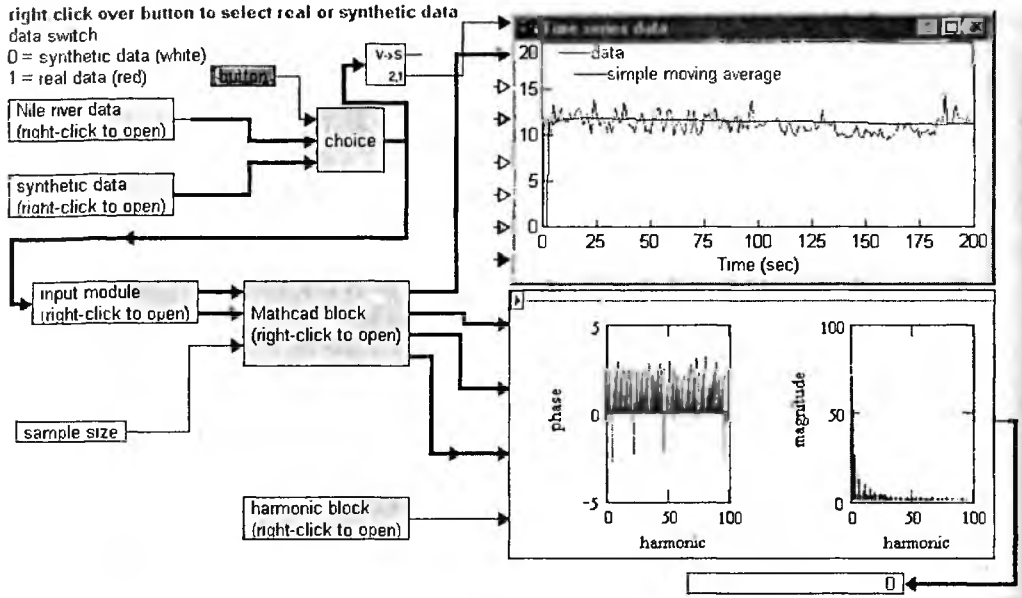


Рис. 6.37. Основная модель анализатора спектра сложных сигналов (анализ реального сигнала)

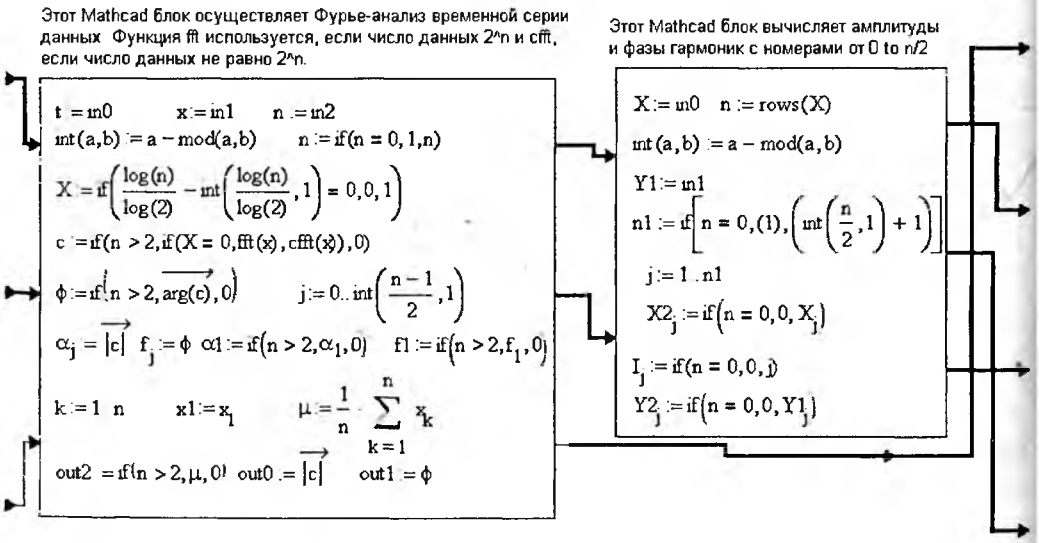


Рис. 6.38. Диаграмма субблока Mathcad block

Первый блок (он показан на рис. 6.38 слева) выполняет Фурье-анализ временного ряда, представляющего входные данные (тестовые или реальные). Интересной особенностью этого блока является автоматический выбор метода спектрального анализа и функций для его реализации. Если число отсчетов данных равно 2n, где n — целое число, то используется наиболее предпочти-

тельная для БПФ функция fft, оперирующая с данными реального типа. Иначе используется функция комплексного преобразования Фурье cfft.

Другой блок (справа) служит для вычисления амплитуд и фаз гармоник с номерами от 0 до n/2 (здесь n уже просто число отсчетов).

Диаграмма субблока ввода исходных данных для проведения спектрального анализа представлена на рис. 6.39. Ввод реализован с помощью линейных регуляторов и не требует ввода каких-либо числовых данных, в которых пользователь может легко ошибиться.

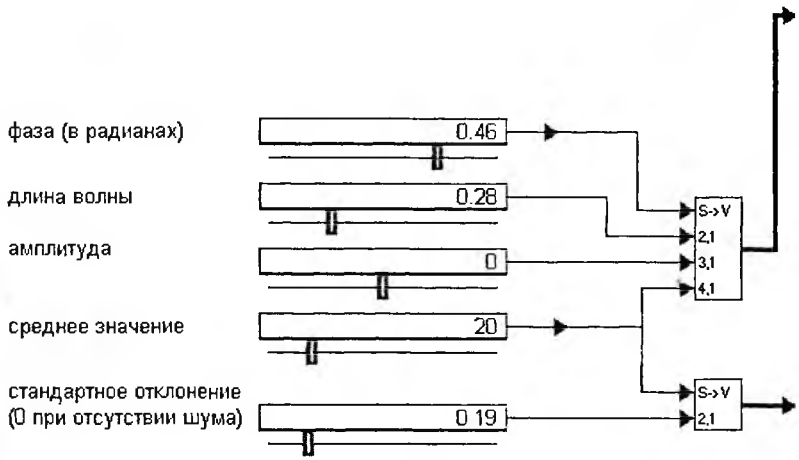


Рис. 6.39. Субблок ввода исходных данных

Остальные блоки особого интереса не представляют. Заинтересованный читатель может познакомиться с ними самостоятельно.

6.4. Интеграция VisSim с матричной системой MATLAB

6.4.1. Интерфейс связи VisSim с системой MATLAB

Как уже отмечалось, система VisSim имеет интерфейс связи и с системой MATLAB. Для его обеспечения нужно при инсталляции VisSim положительно ответить на запрос об интеграции с системой MATLAB и указать путь к файлу matlab.exe. Если это сделано, то в позиции Blocks появится отдел библиотеки блоков MatLab Interface. Этот раздел содержит 4 блока:

1) MatLab Expression — задание выражения, записанного на языке системы MATLAB;

2) MatLab Read Variable — импорт (считывание) значения MATLAB-переменной;

3) MatLab Write Variable — экспорт (запись) в MATLAB переменной;

4) About VisSim/MatLab Interface — вывод окна с информацией о версии интерфейса связи систем VisSim и MATLAB.

Несмотря на небольшое число блоков (из них только первые три предназначены для реальных операций, последний блок выводит чисто информационное окно), они позволяют полноценно использовать вычислительные возможности системы MATLAB, которые признаны одними из самых мощных в решении задач численными методами, и прежде всего относящихся к задачам, связанным с матричными операциями.

6.4.2. Примеры связи VisSim с системой MATLAB

В приведенном на рис. 6.40 примере массив — матрица [1 2; 3 4] присваивается переменной Xmatlab системы MATLAB. Для этого надо ввести этот массив как константу VisSim, а затем ввести блок MatLab Write Variable. В окне его свойств нужно задать имя переменной — Xmatlab, после чего оно и появится в графическом изображении блока, заменив первоначальное имя блока. При исполнении этого примера в рабочем пространстве системы MATLAB будет создана данная переменная с заданным значением. В MATLAB над ней можно будет проделывать любые численные операции, например вычислить детерминант матрицы, осуществить ее транспозицию и инвертирование и т. д.

Этот пример иллюстрирует также считывание переменной Xmatlab из рабочего пространства памяти системы MATLAB. Для этого используется блок MatLab Read Variable, в окне которого (показано на рис. 6.40 снизу) надо задать имя переменной — Xmatlab. Считанное значение индицируется цифровым индикатором системы VisSim. Вы можете задать запуск создания MATLAB переменной или ее считывание только при первом запуске VisSim, точнее, при первом шаге моделирования. Соответствующая опция есть в левом нижнем углу окон свойств переменных.

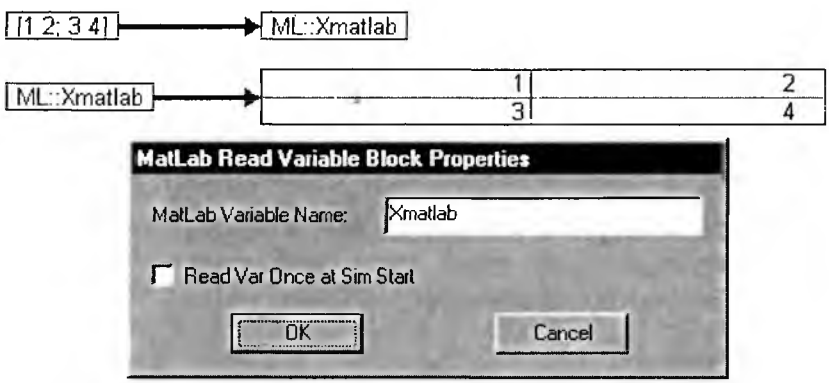


Рис. 6.40. Примеры задания и считывания переменной MATLAB

Все это будет иметь место после пуска моделирования и загрузки системы MATLAB. Работа VisSim была проверена и оказалась успешной при использовании последней (на момент подготовки рукописи книги) версии системы MATLAB 6.5 (реализация 13).

После пуска данных примеров можно обнаружить окно системы MATLAB, в котором будут лишь начальные сообщения:

```
Using Toolbox Path Cache. Type "help toolbox_path_cache" for more info.
To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.
```

Однако нетрудно убедиться в том, что переменная матричного типа Xmatlab действительно создана в рабочем пространстве памяти системы MATLAB. Для этого достаточно исполнить в окне MATLAB команду

```
» Xmatlab
```

Тут же будет выведен следующий результат:

```
Xmatlab =
1  2
3  4
```

Можно также создавать в окне MATLAB новые переменные и вообще использовать любые доступные в этой системе команды. Естественно, что для этого надо ознакомиться с системой с помощью книг [26—27] или иной технической документацией.

Иногда перед использованием MATLAB вместе с VisSim желательно запустить систему MATLAB. О запуске MATLAB можно судить по появлению ярлыка этой программы в окне переключения задач, которое активизируется нажатием клавиш Alt и Tab. Иногда без этого возможно возникновение ошибок, которые следует сбросить командой Clear Errors (или нажатием клавиш Ctrl+E), и пустить модель заново.

6.4.3. Использование в VisSim MATLAB-выражений

Пример на рис. 6.41 показывает применение блока MATLAB-выражения. Поначалу блок создается пустым. Нужное выражение вводится в окне ввода MATLAB-выражения, также показанное на рис. 6.41. После ввода выражение появляется и в самом блоке. Первоначально блок имеет один вход и один выход, но число входов можно увеличивать командой Add Connector. Входы блока рассчитаны на подачу на них векторов или матриц. Внутри блока соответствующие переменные обозначаются как $1, $2 и т. д. Выход блока — один.

Описанный способ использования блока MATLAB-выражений является стандартным или, как принято говорить, документированным. Но он позволяет использовать лишь матричные операции MATLAB, причем матрицы могут быть только численными элементами.

Существует, однако, необычный и недокументированный способ применения блока MATLAB-выражений. Он заключается во включении в данный блок любых MATLAB-выражений при подключении ко входам фиктивных источников, а к выходу — осциллографа или цифрового индикатора. При этом внутри блока можно использовать переменные с любыми именами по правилам, принятым в программировании системы MATLAB.

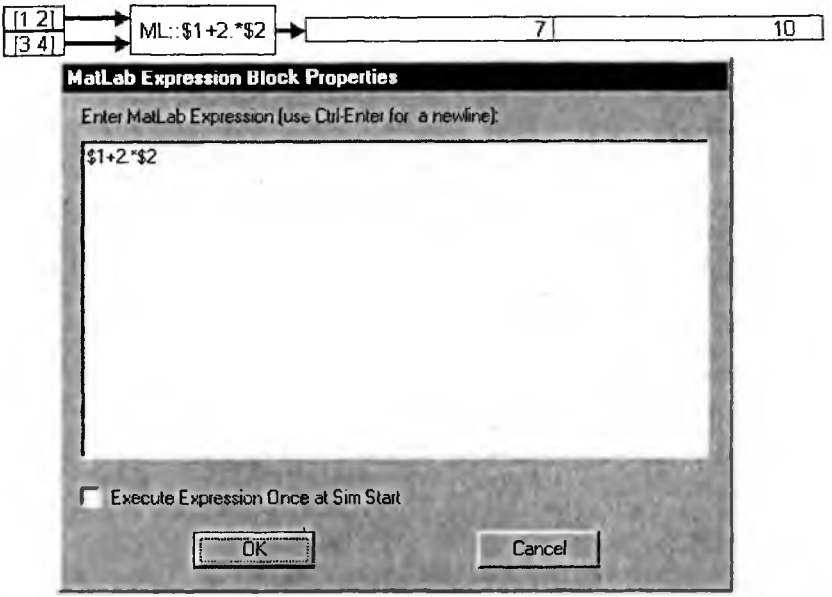


Рис. 6.41. Пример применения блока MATLAB-выражений

Простейший пример такого подхода представлен на рис. 6.42. В данном случае в среде MATLAB к значениям элементов исходной матрицы прибавляется число 10, после чего уже в среде VisSim блоком Display выводится новая матрица. Это и видно внизу рис. 6.42.

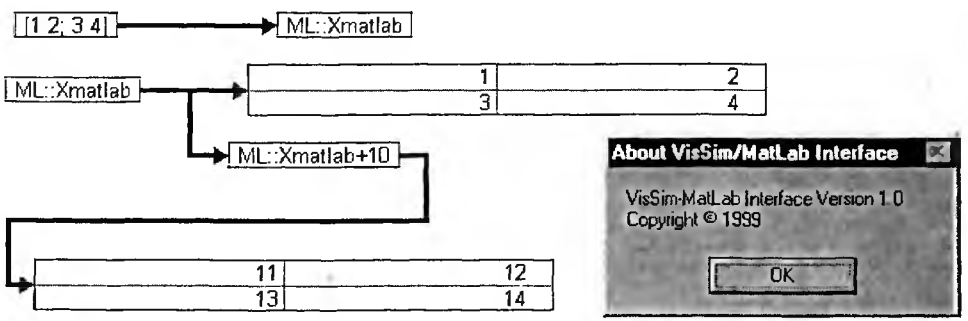


Рис. 6.42. Пример нестандартного применения блока MATLAB-выражений

В правом нижнем углу рис. 6.42 представлено окно, выводимое командой About VisSim/MatLab Interface. Оно дает информацию о версии интерфейса.

6.4.4 Обращение к графическим средствам MATLAB

Система MATLAB имеет уникальную, хотя и довольно специфическую графику. С помощью нестандартного применения блока MATLAB-выражений можно не только выполнять любые вычислительное операции в VisSim моде-

лях, но и обращаться к любым другим, в том числе графическим, средствам системы MATLAB.

Модифицируем приведенный выше пример и добавим в окне свойств MATLAB-выражения строку команд, которые необходимы для построения графиков функции синуса, синуса в кубе и синуса в пятой степени. После ввода команд нажатие клавиши OK приведет к тому, что эти команды появятся внутри блока MATLAB-выражения. Наконец, если навести на этот блок курсор мыши и щелкнуть правой клавишей, то вновь появится окно свойств MATLAB-выражения, что показано на рис. 6.43.

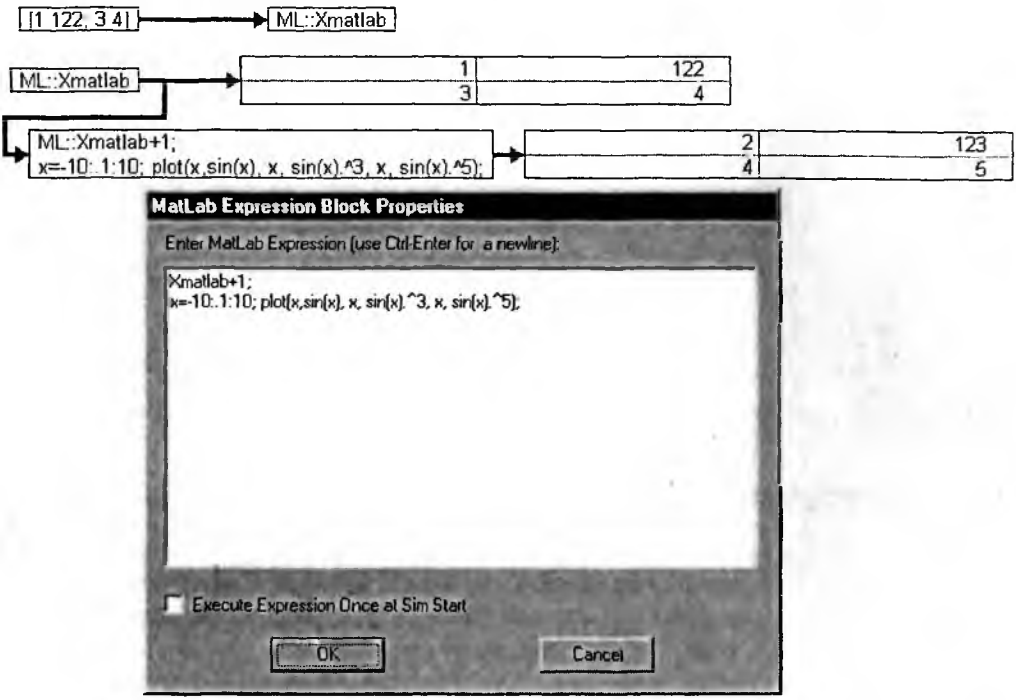


Рис. 6.43. Выполнение моделирование и вывод окна MATLAB-графики с результатами построения графиков трех функций

Если теперь пустить модель на моделирование, то ничего заметного, скорее всего, не произойдет. Однако, нажав клавиши Alt и Tab, можно заметить появление среди ярлыков запущенных приложений еще и ярлыка MATLAB графики. Это связано с тем, что графики MATLAB строит в отдельных перемещаемых и изменяемых в размерах окнах. Можно вывести это окно на переднем плане, что показано на рис. 6.44.

Несмотря на небольшое число интерфейсных блоков для системы MATLAB, последняя благодаря своей универсальности и высокой скорости вычислений способна легко решать задачи, которые с трудом решаются чисто в среде VisSim.

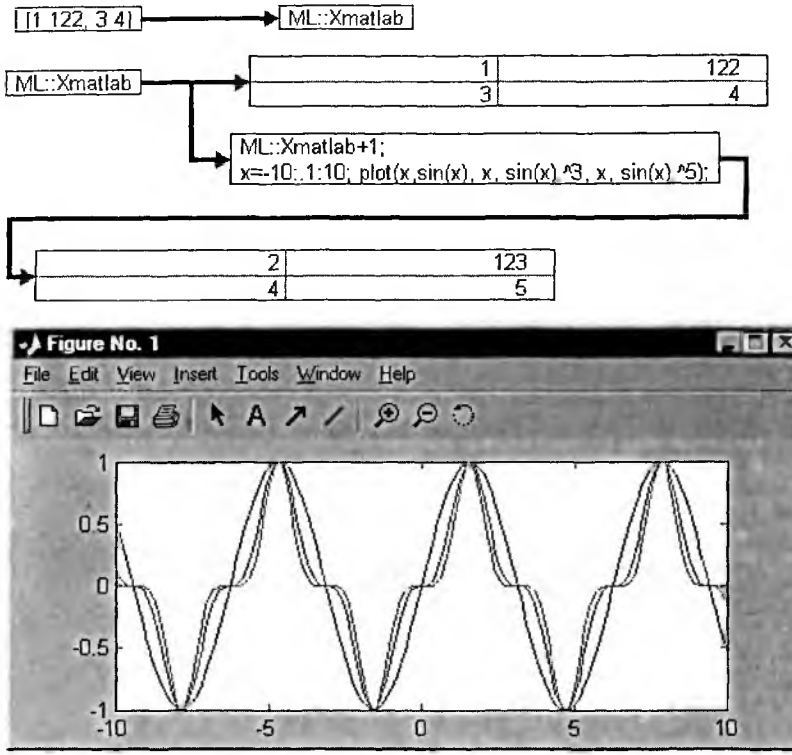


Рис. 6.44. Пример использования графических средств системы MATLAB

6.4.5. Моделирование временных характеристик линейной системы

MATLAB и его пакеты расширения имеют уникальный набор функций для моделирования линейных и нелинейных систем различного вида. В качестве примера рассмотрим моделирование переходных процессов для следующей модели 2-го порядка:

$$y_k = \sum_{i=0}^{N-1} (x1_{k-i} \cdot x2_i).$$

В MATLAB для этого достаточно в командном режиме использовать следующие команды:

```
» a=[-0.5572 -0.7814;0.7814 0];
» b=[1 -1;0 2]; c=[1.9691 6.4493];
» sys=ss(a,b,c,0); step(sys)
```

Модель VisSim, решающая данную задачу, представлена на рис. 6.45. С помощью блоков VisSim задаются массивы a, b и c в виде MATLAB-переменных. Блок выражений изначально приспособлен для вычисления отдельных выражений. Поэтому для его использования надо (хотя бы фиктивно) задействовать его входы и выходы. В нашем случае на входы поданы перемен-

ные MATLAB, а выход просто передает значение матриц на цифровой индикатор. Однако главный продукт работы этой модели в ином. Модель создает систему sys с помощью функции создания передаточной функции ss и затем вычисляет с помощью функции step(sys) переходные характеристики по двум параметрам и выводит в отдельное окно их графики.

Заменив функцию step(sys) на impulse(sys), можно смоделировать и построить графики импульсных характеристик, приведенные на рис. 6.46.

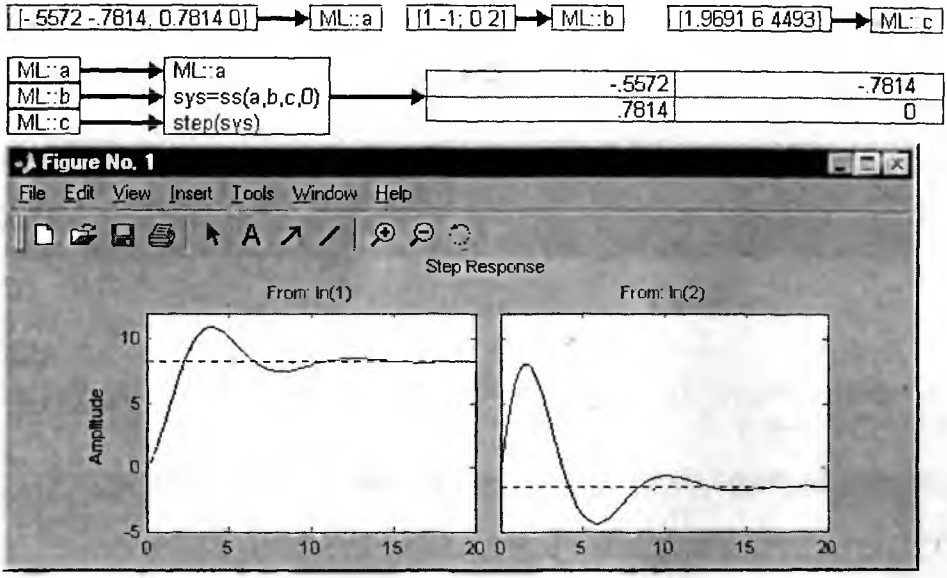


Рис. 6.45. Построение переходных характеристик системы в среде VisSim+MATLAB

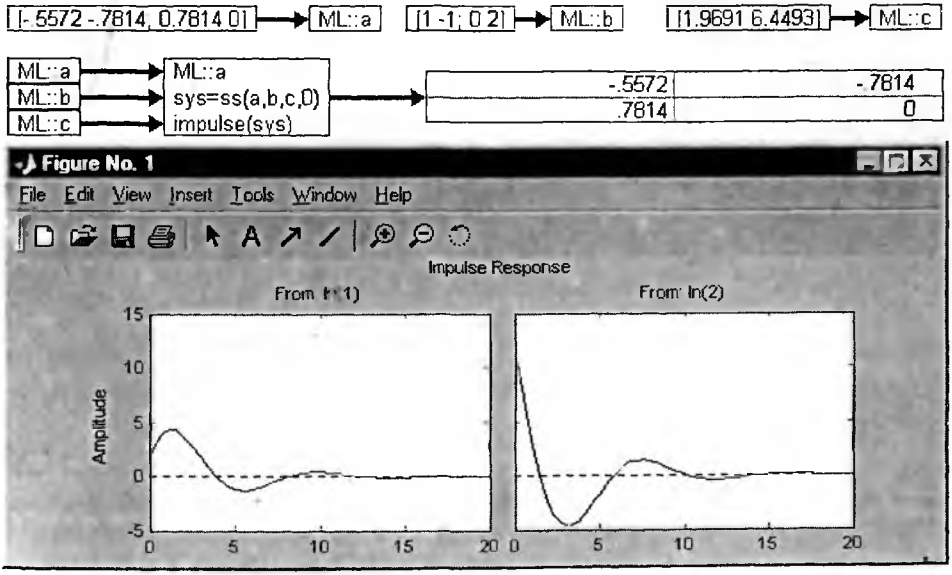


Рис. 6.46. Построение импульсных характеристик системы в среде VisSim+MATLAB

Любопытно, что построить их одновременно данная модель не позволяет. Видимо, это одно из ограничений нестандартного применения блока MATLAB.

6.4.6. Построение диаграммы Николса

Для анализа поведения и устойчивости систем применяют диаграммы Найквиста и Николса. Диаграмму Найквиста позволяет легко строить приложение Analizy системы VisSim, так что остановимся на более информативной диаграмме Николса, построение которой блоками VisSim не предусмотрено, но возможно средствами системы MATLAB.

Рисунок 6.47 показывает модель для построения диаграммы Николса по передаточной характеристике, заданной коэффициентами числителя num и знаменателя den. Эти векторы заносятся в рабочее пространство как MATLAB-переменные. Построение диаграммы Николса задано в блоке MATLAB-выражения.

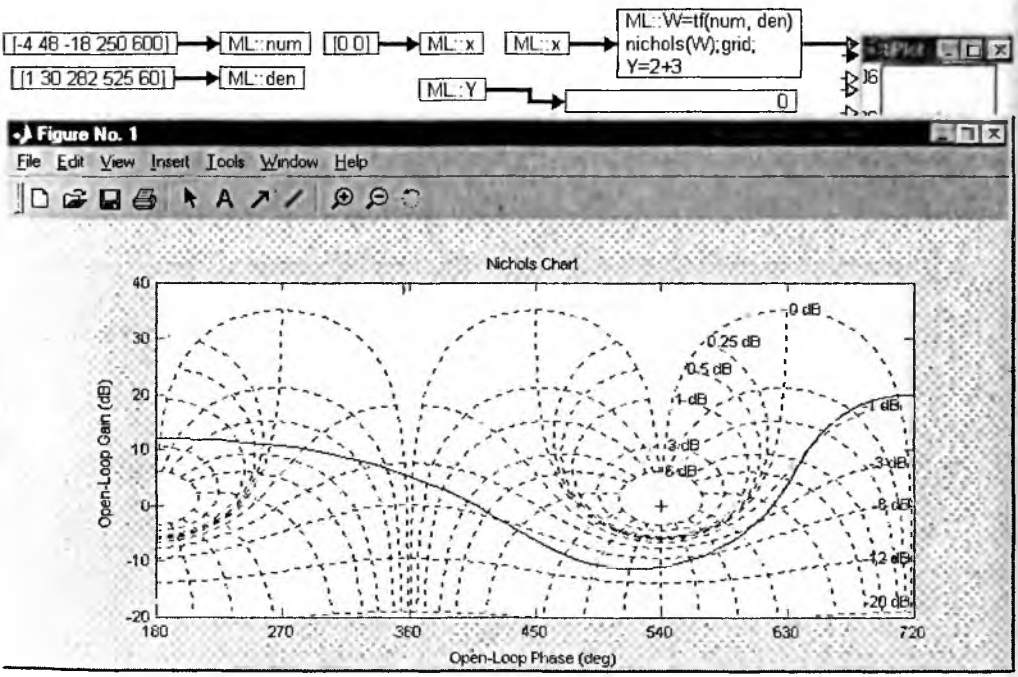


Рис. 6.47. Анализ линейной системы, заданной передаточной характеристикой, и построение диаграммы Николса

Эту диаграмму, как и другие графики системы MATLAB, можно менять в размерах, вводить в нее различные комментарии и использовать весь набор средств форматирования графиков, которыми обладают графические окна этой системы.

6.4.7. Ограничения нестандартного применения блока MATLAB-выражений

Описанное выше нестандартное применение блока MATLAB-выражений заметно расширяет возможности совместного применения объединенной системы VisSim и MATLAB. Однако, как любое нестандартное применение, оно не гарантирует безупречную работу такой системы. Наряду с локальными ошибками (индицируются красным цветом блоков и вызывают остановку моделирования) возможны и более опасные ситуации вплоть для фатальных системных ошибок, сбрасывающих VisSim (рис. 6.48), и зависания ПК.

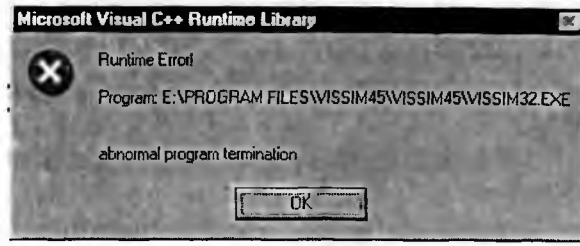


Рис. 6.48. Окно фатальной ошибки, сбрасывающей VisSim

Во избежание потери созданной модели перед ее пуском рекомендуется сохранить ее в файле. Новый запуск моделирования нужно проводить после тщательного изучения причин возникновения серьезных ошибок. Использовать блок MATLAB-выражения в нестандартном применении стоит только в исключительных случаях и, как правило, с единичным запуском (с параметрами моделирования Start = 0, Step = 1 и End = 1). Вы можете задать запуск MATLAB только на первом шаге моделирования, включив опцию Execute Expression Once at Sim Start в окне свойств блока.

И наконец, самое главное. Система MATLAB — это монстр среди систем компьютерной математики. Новая реализация MATLAB 6.5 при полной установке занимает около 1500 Мб, тогда как VisSim довольствуется несколькими мегабайтами. Впрягать MATLAB в VisSim примерно тоже самое, что в «Жигули» установить авиационный двигатель с «Боинга». Что будет после его запуска, очень напоминает то, что может произойти с VisSim, — авария! К тому же MATLAB имеет свое встроенное приложение для блочного имитационного моделирования — Simulink. Так что о технической целесообразности совместного применения VisSim и MATLAB стоит призадуматься. Но из песни слов не выкинешь — в принципе оно возможно.

6.5. Работа с расширением VisSim/Comm

6.5.1. Назначение пакета расширения VisSim/Comm и состав его библиотеки

Пакет расширения VisSim/Comm — это специальное средство для моделирования и проектирования самых разнообразных связных, коммуникационных и радиотехнических устройств. Простейший тракт связи может быть представлен следующим образом:

Сигнал → Кодировщик → Модулятор → Канал → Демодулятор → Декодер → Сигнал

Первые три части тракта представляют передатчик информации, а последние — приемник информации. Они соединены каналом связи. Разумеется, в тракт могут быть введены и другие элементы, например ослабители сигналов (аттенюаторы), компрессоры данных, эквалайзеры, различные фильтры и иные устройства.

Пакет VisSim/Comm 5 содержит богатую библиотеку блоков, содержащую свыше 170 блоков. В 15 разделах библиотеки расширения VisSim/Comm содержится:

1) Channels — 11 блоков построения каналов связи;
2) Complex Math — 11 блоков с операциями с комплексными данными (числами);
3) Demodulators — 6 блоков демодуляторов;
4) Digitals — 15 блоков построения цифровых устройств общего назначения;
5) Encode/Dcode — 15 блоков создания кодирующих и декодирующих устройств;
6) Estimators — 12 блоков оценивания;
7) Filters — 11 блоков построения цифровых фильтров;
8) Instruments — 4 блока инструментов;
9) Modulators-Complex — 12 блоков построения модуляторов на основе аппарата комплексных чисел;
10) Modulators-Real — 11 блоков построения модуляторов на основе аппарата действительных чисел;
11) Multirate Support — 3 блока ;
12) PLL — 6 блоков ;
13) RF — 7 блоков приемных устройств;
14) Signal Sources — 16 блоков источников сигналов;
15) Vector Ops — 8 блоков векторных операций.

В папке Comm Examples содержится множество демонстрационных примеров на применение пакета VisSim/Comm. При инсталляции пакета, помимо новой позиции Comm меню системы VisSim, появляется еще одна новая позиция — Wireless (Беспроводная связь). Она позволяет включать в модели VisSim блоки модуляторов беспроводной связи в соответствии со стандартами 802.11 (варианты GFSK-2 и GFSK-4) и Bluetooth.

Полное и детальное описание пакета расширения VisSim/Comm требует объема, заметно большего, чем объем всей этой книги, поскольку средства пакета охватывают большие разделы по классическим и самым современным цифровым методам обработки сигналов. Учитывая это, а также то, что пакет VisSim/Comm пока распространен гораздо меньше, чем сама система VisSim 4.5/5, мы рассмотрим средства этого пакета обзорно, на основе ограниченного, но отнюдь не малого числа примеров и без детального описания всех блоков библиотеки. Такое обзорное описание позволит пользователю познакомиться с возможностями данного расширения, а заинтересованный пользователь может познакомиться самостоятельно с возможностями пакета по его справке и множеству демонстрационных примеров.

6.5.2. Расширенные операции с комплексными числами

Радиотехническая и связная направленность средств пакетов VisSim/Comm потребовала дальнейшего усиления средств для операций с комплексными числами и данными, основанными на применении таких чисел. Соответствующие блоки сосредоточены в разделе библиотеки Complex Math. Рисунок 6.49 демонстрирует применение основных блоков из этого раздела библиотеки.

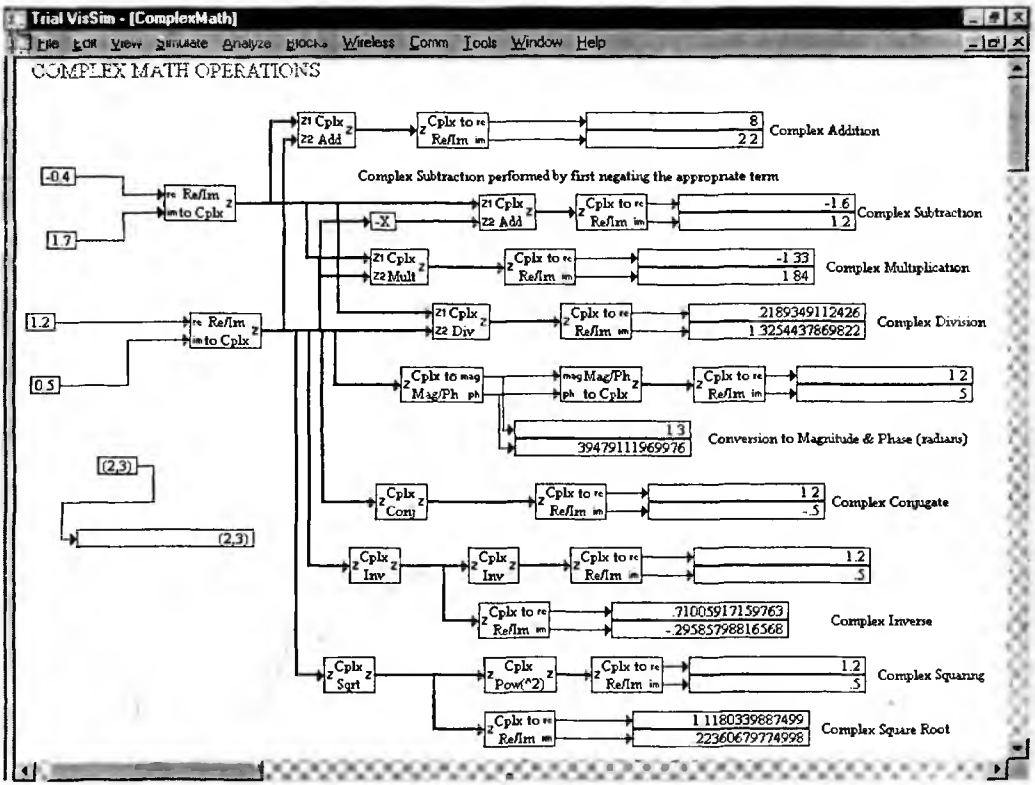


Рис. 6.49. Применение блоков операций с комплексными числами

6.5.3. Новые источники сигналов

В разделе библиотеки Signal Sources имеется весьма представительный набор из 16 источников (виртуальных генераторов) сигналов. Наиболее простые из них представлены на рис. 6.50. Это генераторы следующих самых распространенных сигналов: симметричных прямоугольных импульсов (меандра), треугольных колебаний и пилообразных импульсов. Все источники содержат помимо основного выхода выход синхронизирующих коротких импульсов.

Окно свойств источников импульсных сигналов показано на рис. 6.51. Предусмотрена установка вполне очевидных параметров этих источников, а также выбор формы сигнала в правой части окна.

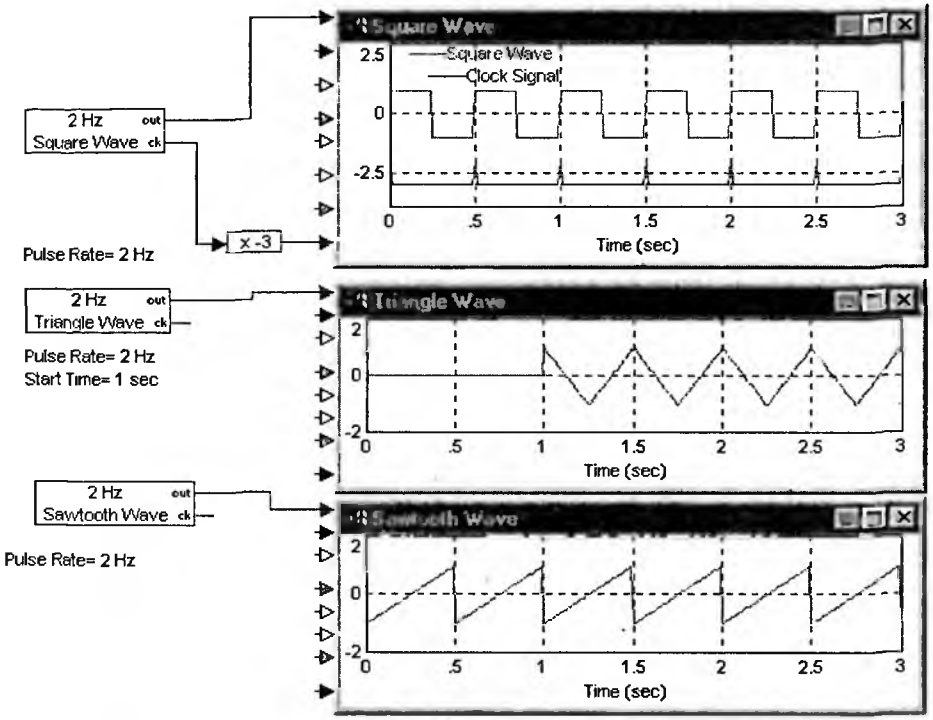


Рис. 6.50. Примеры применения импульсных сигналов

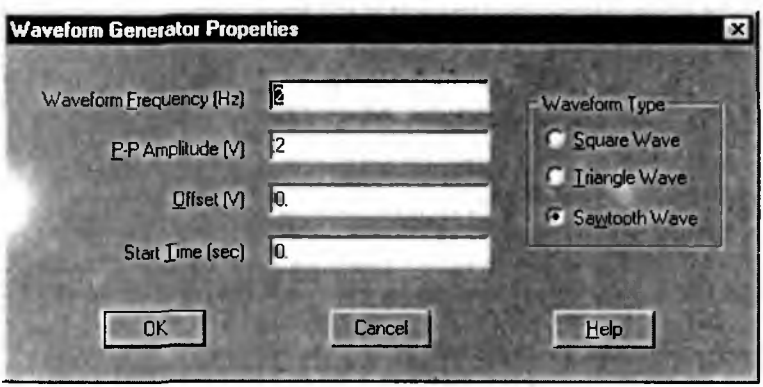


Рис. 6.51. Окно свойств источников импульсных сигналов

Имеется возможность задания сигнала по данным, которые считываются из файлов. Примеры этого представлены на рис. 6.52.

Для снятия АЧХ и ФЧХ различных устройств, например усилителей и фильтров, широко используются генераторы качающейся частоты, именуемые также свип-генераторами. Блок Freq Sweep имитирует работу такого устройства. Его простейшее применение и вид сигнала представлены на рис. 6.53.

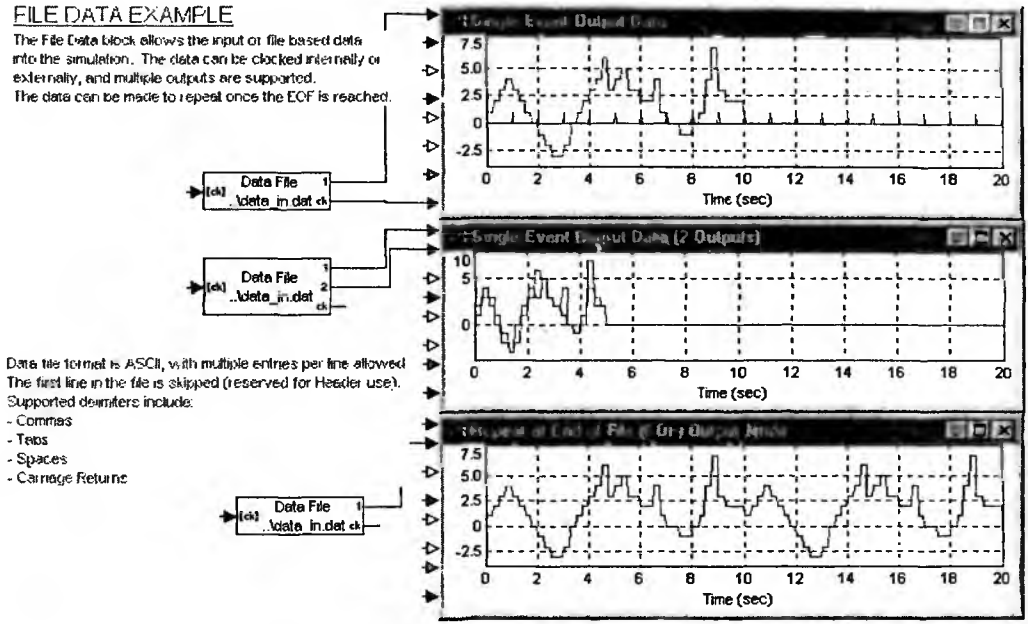


Рис. 6.52. Примеры задания сигналов по данным, считываемым из файлов

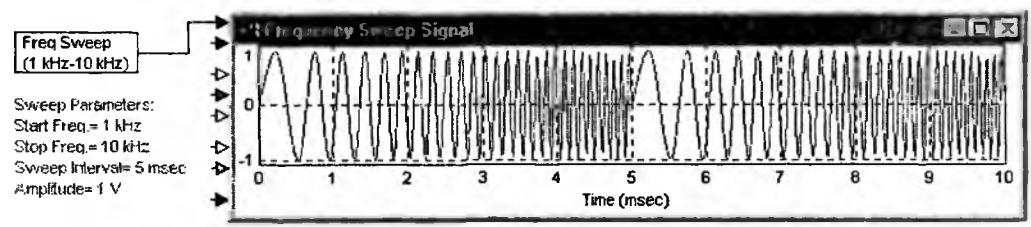


Рис. 6.53. Применение блока Freq Sweep

Для создания импульсных всевдослучайных последовательностей служит блок PN Sequence. Примеры его применения представлены на рис. 6.54. Блок имеет вход для импульсов синхронизации (времени). Он может быть не задействован, и тогда синхронизация происходит с периодом моделирования. Блок имеет два выхода — один out для создаваемой последовательности и второй ck для синхронизирующих импульсов.

Окно свойств блока импульсных последовательностей показано на рис. 6.54. Установки блока достаточно очевидны. Стоит отметить, что число импульсов в последовательности задается в восьмеричной системе.

Для выделения полезных спектральных составляющих используется блок спектрального маскирования Spectral Mask. Применение этого блока представлено на рис. 6.56. Форма кривой маскирования задается из файла, и ее представление в виде таблицы представлено на рис. 6.56. Там же показан блок задания спектрограммы сигнала.

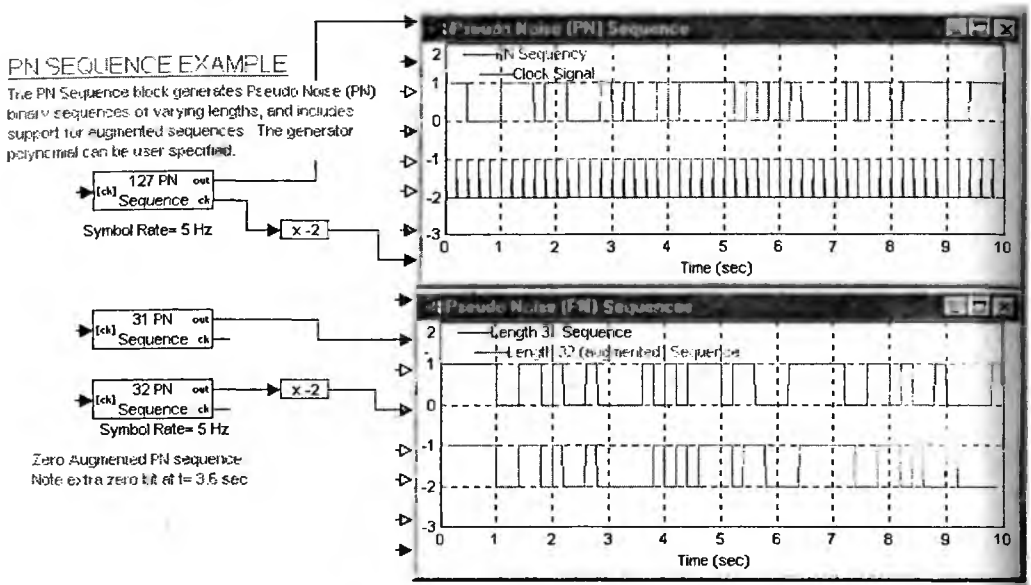


Рис. 6.54. Создание импульсных последовательностей

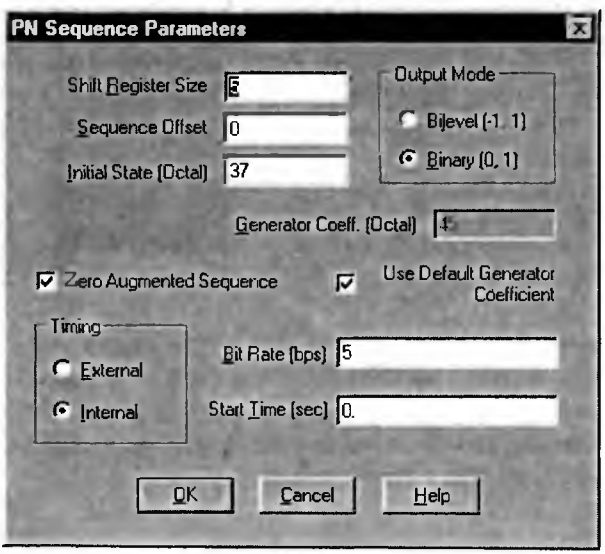


Рис. 6.55. Окно свойств импульсных последовательностей

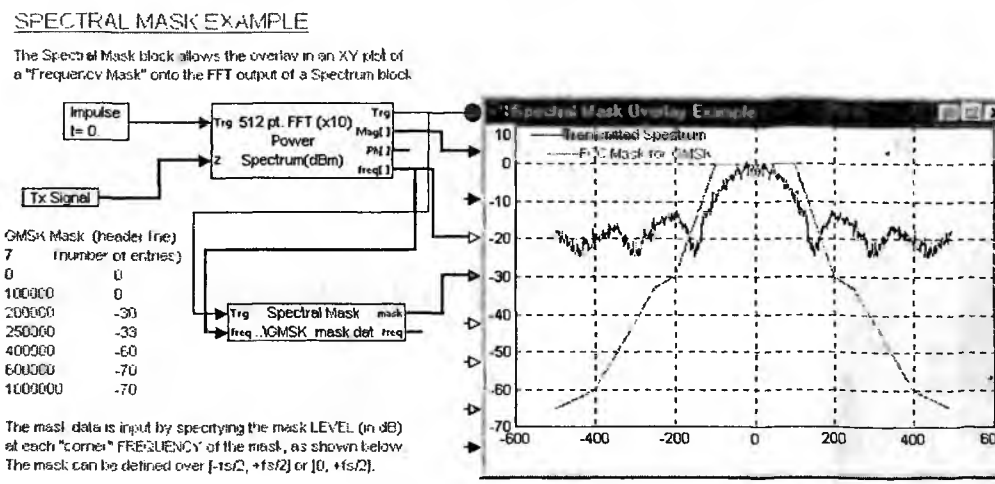


Рис. 6.56. Применение блока Spectral Mask

6.5.4. Кодирование/декодирование сигналов

В разделе библиотеки Encode/Decode полтора десятка блоков для реализации операций кодирования/декодирования. Остановимся на трех наглядных примерах.

На рис. 6.57 представлена модель тракта передачи информации с применением кодера на основе операции свертки. Он реализован блоком Convolutional Encoder. Созданная бинарная последовательность поступает на вход бинарного симметричного канала (блок BSC). Для декодирования данных используется блок Viterbi Hard (в библиотеке есть и Viterbi Soft).

Для кодирования используются различные коды, например Рида—Соломона и Грея. Блоки для их реализации также представлены в разделе библио-

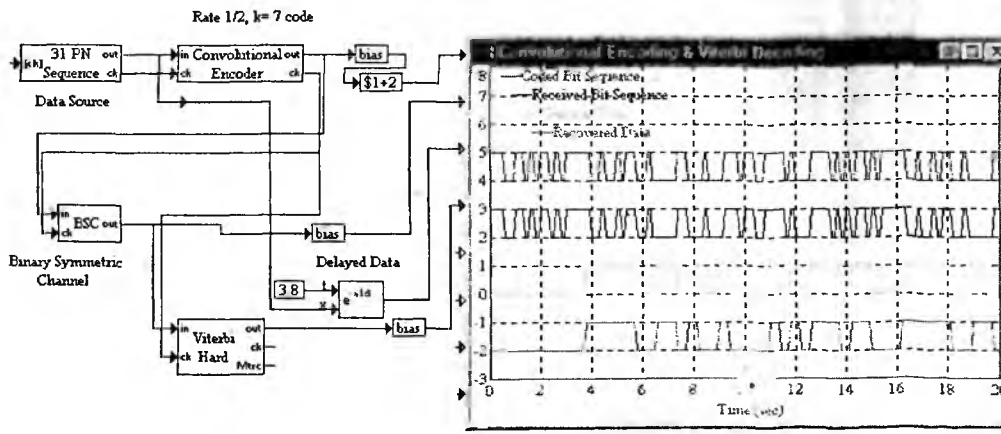


Рис. 6.57. Пример построения симметричного бинарного канала с кодером на основе свертки и декодером типа Viterbi Hard

теки Encode/Decode. Рисунок 6.58 показывает результат кодирования линейно нарастающего квантованного сигнала (верхняя осциллограмма) кодом Грея (нижняя осциллограмма) и затем результат декодирования с применением обратного кода Грея (под осциллограммой сигнала). Нетрудно заметить, что (с учетом небольшого смещения осциллограмм входного сигнала и сигнала на выходе блока обратного кода Грэя) входной и выходной сигналы являются идентичными.

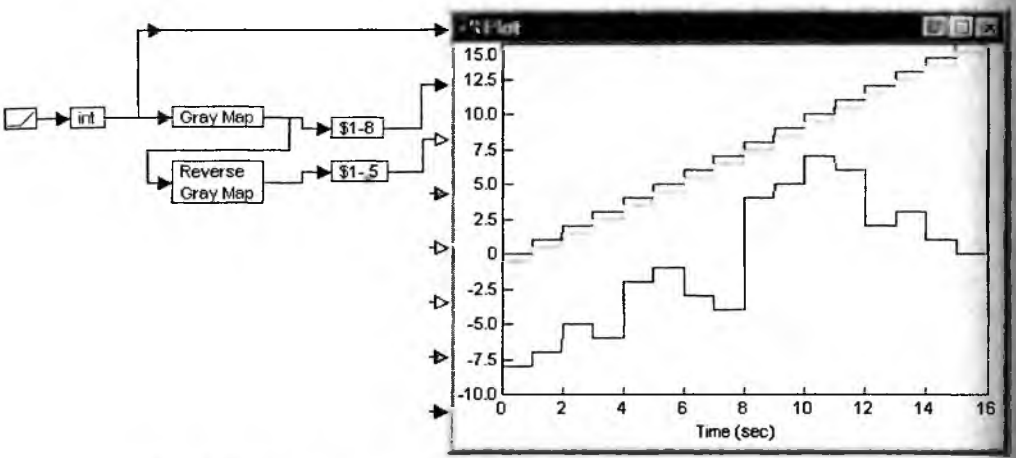


Рис. 6.58. Моделирование кодирования/декодирования кодом Грэя

Еще одним примером моделирования кодирования/декодирования является модель модема V32, представленная на рис. 6.59. Здесь кодируется случайный импульсный сигнал от блока Random Sys. Используется блок дифференциального кодера V32 Differential Encoder. Затем с применением блоков Trellis Encoder, Distortion, Detector и Trellis Decoder имитируется телефонная

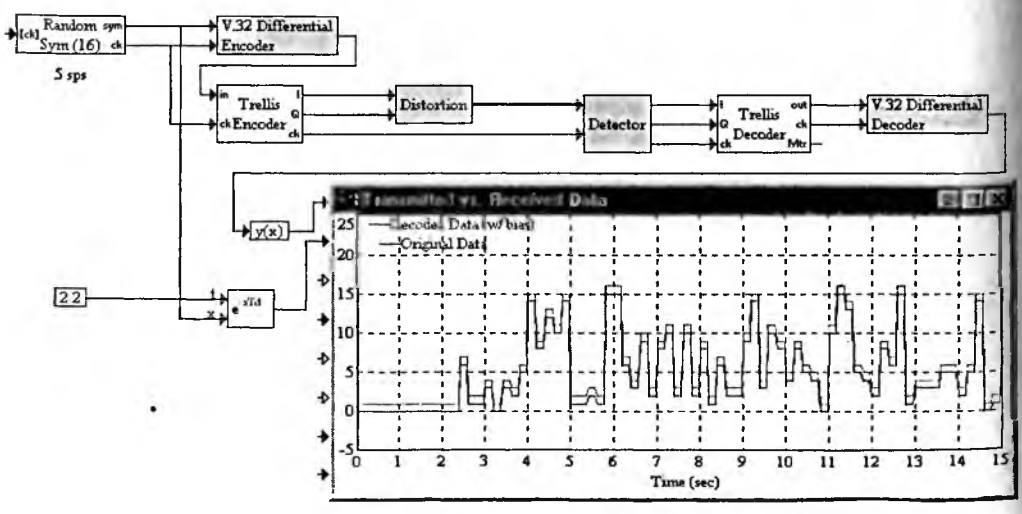


Рис. 6.59. Моделирование модемной линии передачи данных

линия связи с заметным затуханием и частотными искажениями. После декодирования блоком Differential Decoder сигнал полностью восстанавливается.

6.5.5. Модуляторы и демодуляторы

К числу важнейших блоков в трактах связи относятся и модуляторы. Они используются, в частности, в беспроводных линиях связи, когда нужную информацию требуется «наложить» на какой то сигнал, чаще всего высокочастотный синусоидальный. Используются различные виды модуляции — от широко распространенных амплитудной и частотной до однополосной модуляции, создающей сигналы с одной боковой частотой и подавленной несущей. Блоки модуляторов расположены в разделах библиотеки VisSim/Comm Modulators Complex и Modulators Real, а демодуляторы в разделе Demodulators.

Рисунок 6.60 иллюстрирует осуществление амплитудной модуляция с помощью блока AM mod осуществляется модуляция синусоидального сигнала входным сигналом более сложной формы и с более низкой частотой. Демодуляция осуществляется с применением синхронного детектора (блок Synchronous Det.) и пикового детектора (блок Peak Rectifier) с фильтром LPF. Оба детектора неплохо восстанавливают входной сигнал.

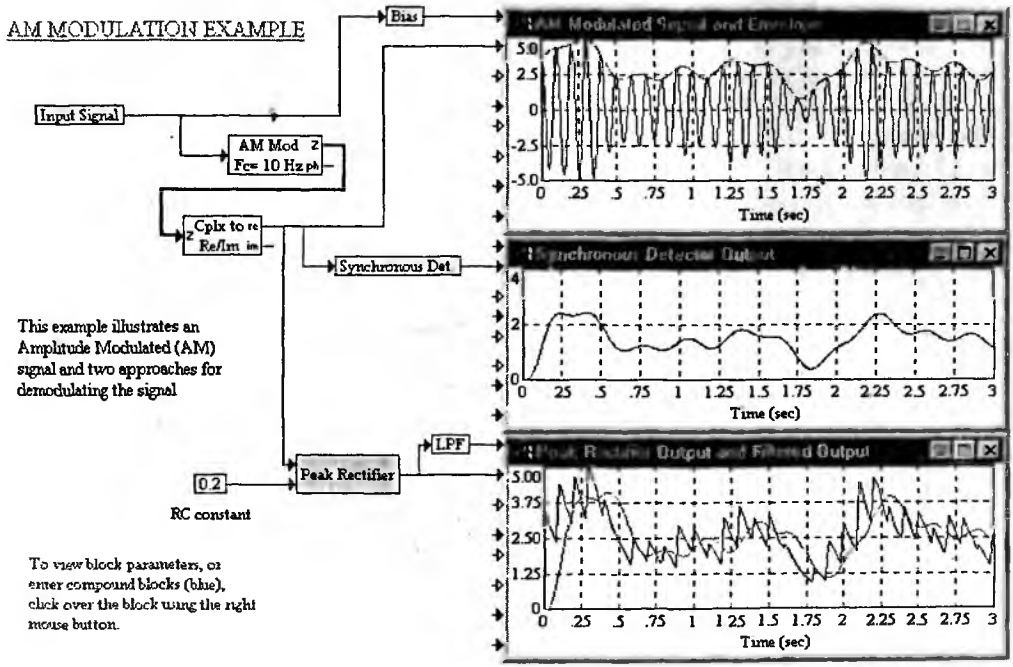


Рис. 6.60. Амплитудная модуляция и демодуляция

Моделирование частотной модуляции/демодуляции демонстрирует рис. 6.61. Частотную модуляцию синусоидального сигнала со средней частотой 3 кГц здесь моделирует блок FM Mod. А блок FM Demod используется для демодуляции и восстановления исходного сигнала.

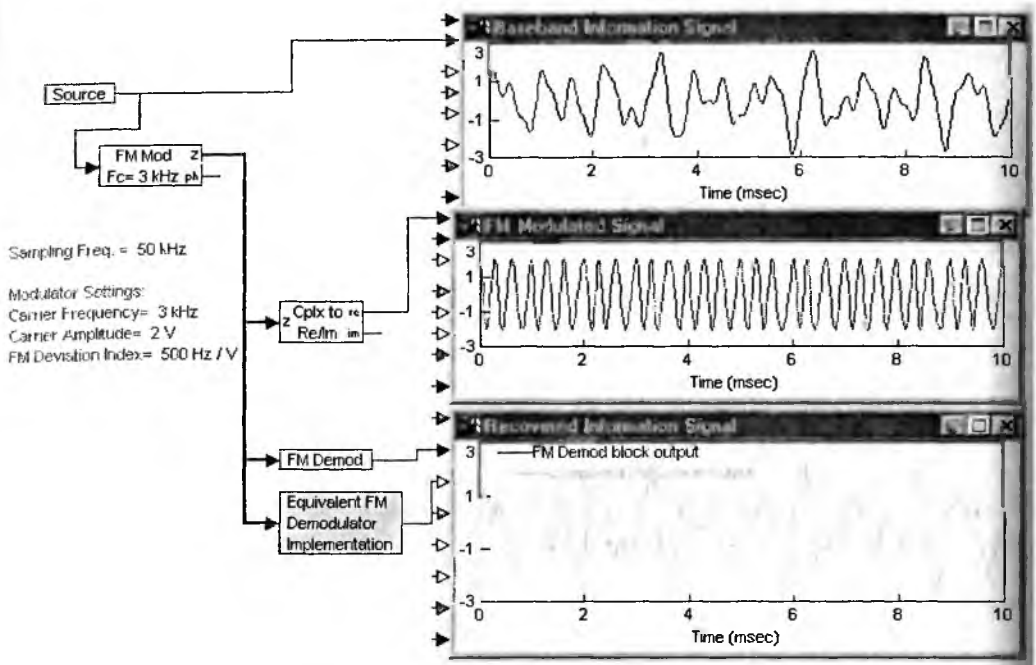


Рис. 6.61. Моделирование частотной модуляции и демодуляции

Кроме того, в этой модели имеется субблок Equivalent FM Demodulator Implementation, демонстрирующий процесс демодуляции частотно-модулированного колебания. Диаграмма этого субблока представлена на рис. 6.62.

В библиотеке VisSim/Comm имеется также ряд блоков для осуществления различных видов импульсной и кодоимпульсной модуляции. Рисунок 6.63 демонстрирует реализацию широтно-импульсной модуляции с помощью блока PPM Mod. На вход блока подается случайный сигнал в виде 4-уровневых ступеней. Он создается блоком Random sys. Для демодуляции используется блок PPM Demod.

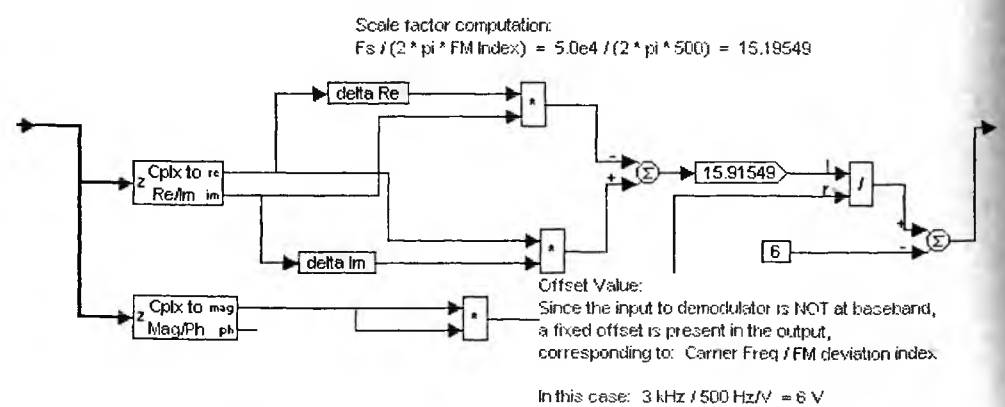


Рис. 6.62. Субблок Equivalent FM Demodulator Implementation

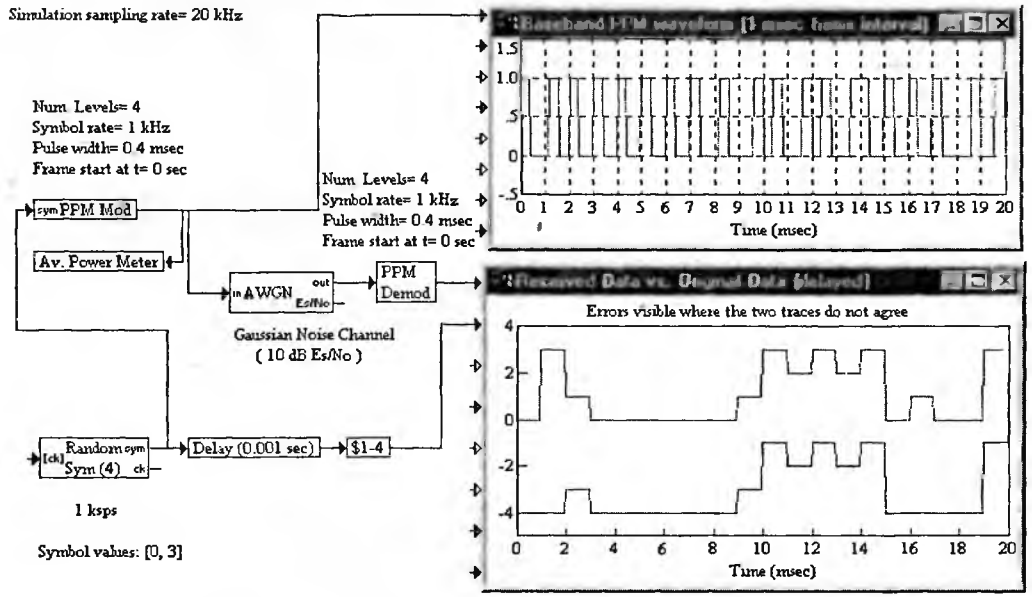


Рис. 6.63. Пример реализации широтно-импульсной модуляции

К сожалению, в данном случае идеального восстановления сигнала нет. В этом можно убедиться, сравнивая осциллограммы сигнала на входе модулятора и на входе демодулятора.

6.5.6. Организация каналов

В разделе библиотеки Chanels размещены блоки, полезные для организации каналов связи с различными видами передаваемой по каналам информации. Некоторые из блоков этой библиотеки уже были рассмотрены.

На рис. 6.64 показана организация канала связи по 4 каналам с помощью блока Multipath Canel (Канал с множеством путей). Поясняет работу этого блока его имитация с помощью блоков временной задержки и блоков задания коэффициентов передачи. На рис. 6.64 показано также окно свойств блока Multipath Canel. Полезно сравнить его установки с установками времен задержки и коэффициентов передачи в нижнем примере организации четырехканального тракта передачи данных.

Часто возникает необходимость по одному каналу передавать несколько разных сигналов. Для временного уплотнения сигналов в один канал используются мультиплексоры, а для обратного действия — демультиплексоры. Рисунок 6.65 демонстрирует применение блоков мультиплексора Mux и демультиплексора Demux для передачи по одному каналу четырех различных сигналов в виде синусоид с разными частотами и амплитудами.

Нередко необходимо учитывать затухание в канале связи. Для этого можно использовать управляемый блок затухания Prop Loss. Его применение представлено на рис. 6.66. Блок имеет вход сигнала in и вход задания дистан-

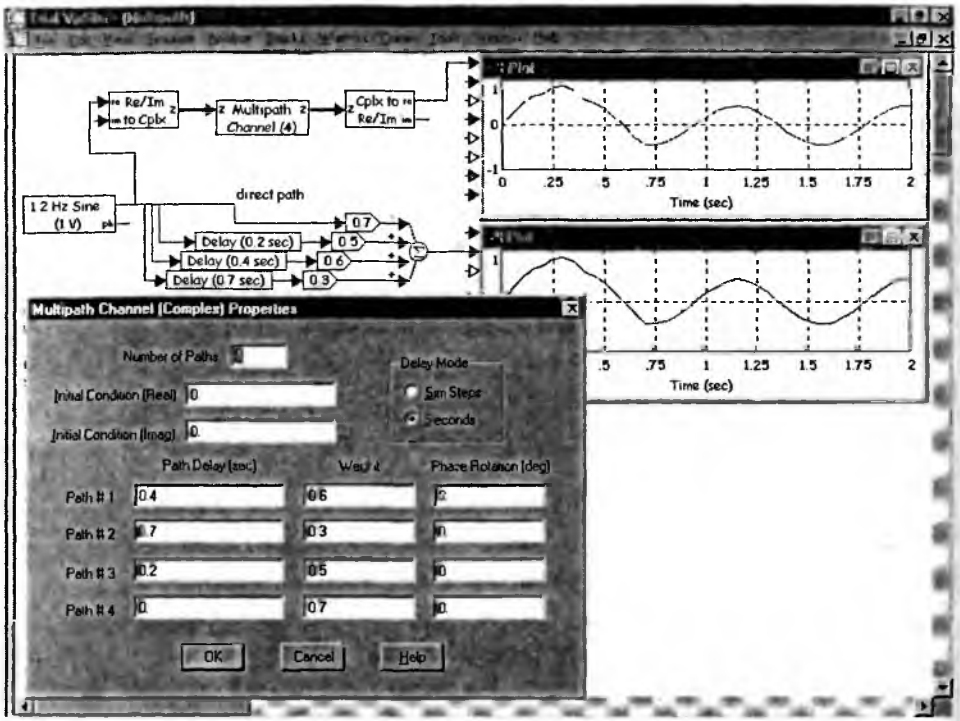


Рис. 6.64. Пример применения блока Multipath Canel

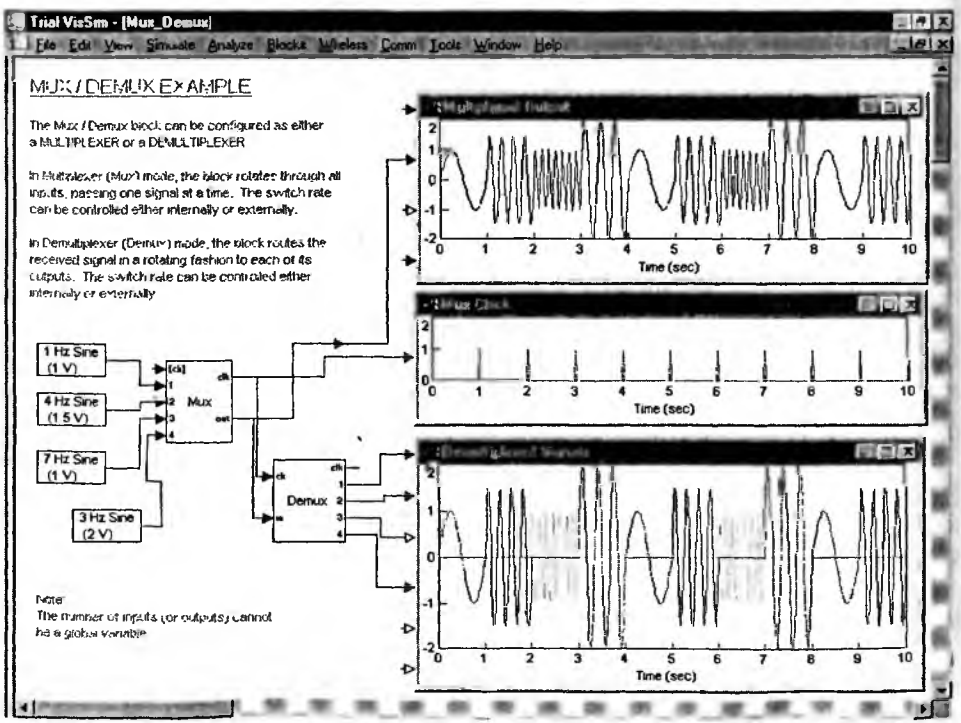


Рис. 6.65. Применение мультиплексора и демультиплексора

ослабление в децибелах. Модель рис. 6.66 имитирует затухание сигнала по мере его продвижения по каналу связи.

А в заключение этого раздела рассмотрим еще один канал связи на основе кодера и декодера Рида—Соломона (RS) и блока Vector AGWN. Модель такого канала связи представлена на рис. 6.67. На вход кодера подается последова-

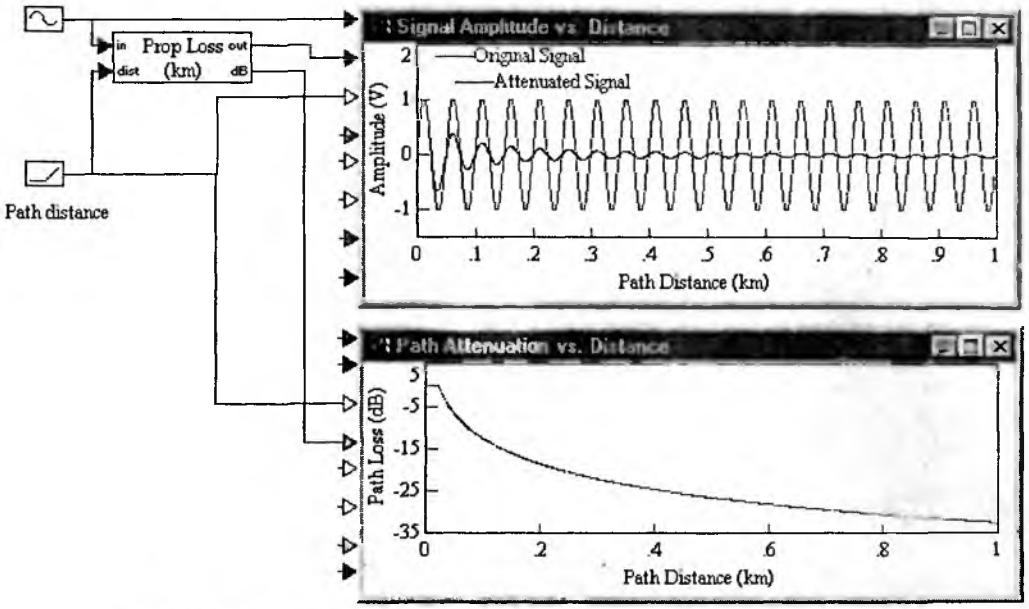


Рис. 6.66. Моделирование затухания в канале связи в функции от расстояния

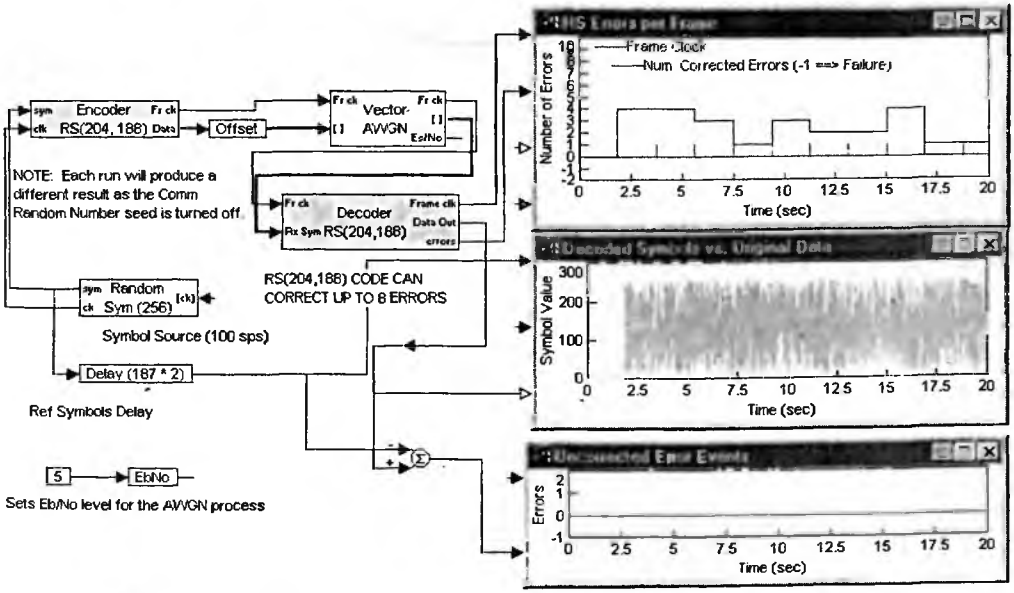


Рис. 6.67. Моделирование канал связи на основе кодера и декодера Рида—Соломона

тельность кодов в виде случайных чисел с кодами от 0 до 255. Они имитируют передачу текстовых данных в формате ASCII.

В этой модели контролируется появление исправляемых с помощью кодов Рида—Соломона ошибок (верхняя осциллограмма), поток данных (средняя осциллограмма) и результирующая ошибка (нижняя осциллограмма). Как нетрудно отметить, кодер/декодер на основе кодов Рида—Соломона прекрасно справляется с коррекцией ошибок, так что результирующая погрешность равна 0 в течение всего времени моделирования.

6.5.7. Применение блоков оценивания параметров

В разделе библиотеки Estimators представлена дюжина блоков оценки параметров. Рассмотрим их начиная с простого блока оценки времени задержки Delay Estimator. Блок имеет два входа и два выхода. На вход in подается задержанный, а на вход ref опорный сигнал. На выходе out формируется время задержки, а на выходе flag сигнал для индикации текущего состояния блока (идут вычисления или нет). Рисунок 6.68 показывает применение блока.

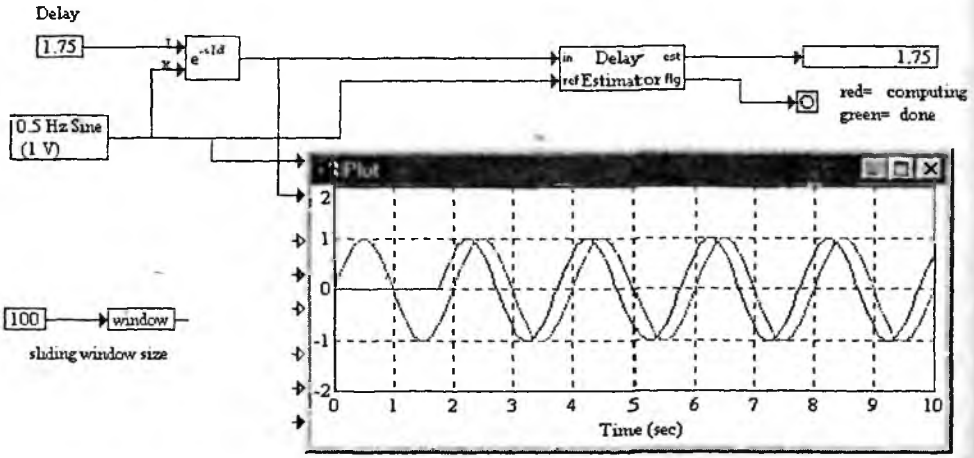


Рис. 6.68. Оценивание времени задержки

Блок оценки времени Time when служит для вычисления времени от начала моделирования и до момента, когда сигнал на входе блока начнет удовлетворять записанному в нем неравенству. Рисунок 6.69 показывает вычисление моментов времени на выходе НЧ-фильтра Баттерворта с шумовым сигналом на входе для условий, когда уровень сигнала достигает и превышает значение 1 и когда он достигает значения –1.2 и становится меньше его.

Для оценки автокорреляции между двумя сигналами служит блок Sliding Correlator (Следящий коррелятор). Его применение поясняет рис. 6.80. Здесь с помощью двух FIR-фильтров созданы сдвинутые во времени отклики фильтров. Они подаются на входы in и ref (сигнальный и опорный). Осциллограмма на выходе блока также представлена на рис. 6.70.

Функции блока автокорреляции можно пояснить следующим образом. Пусть есть три сигнала: x1 — входной сигнал, x2 — опорный сигнал и x3 —

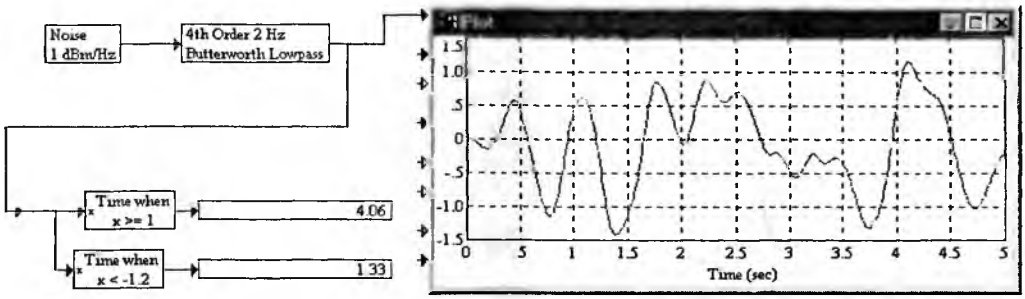


Рис. 6.69. Оценка времен достижения сигналом заданных значений

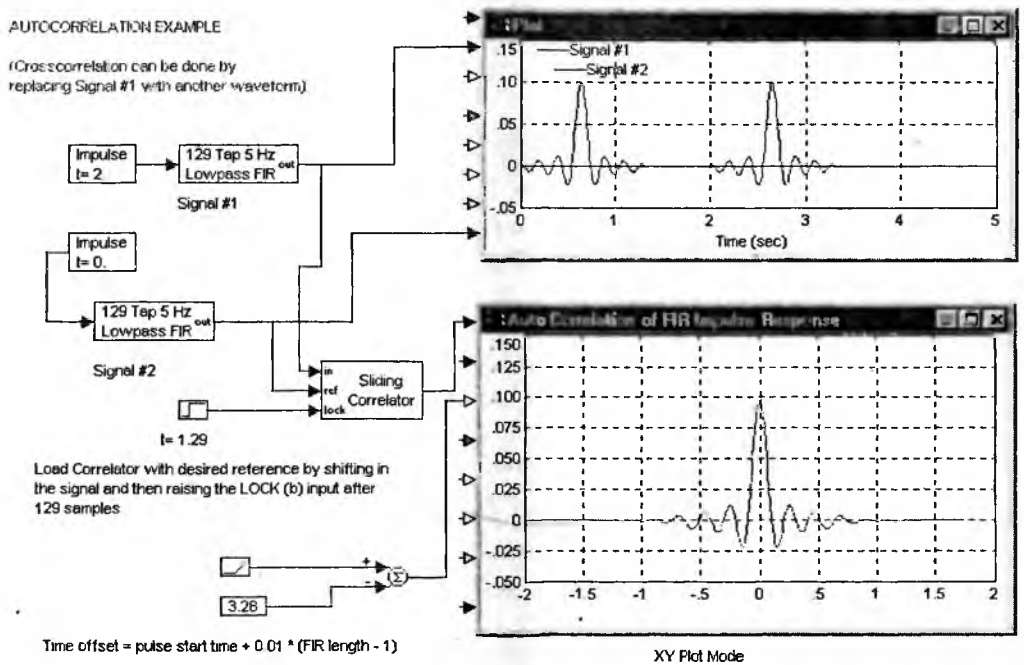


Рис. 6.70. Пример применения блока автокорреляции

сигнал синхронизации (0 или 1, при 1 пуск блока). Того на выходе блока следящей корреляции будет создан сигнал y, который описывается следующим выражением:

$$y_k = \sum_{i=0}^{N-1} (x1_{k-i} \cdot x2_j),$$

где $j = k - i$ в следящем режиме (sliding mode) и $j = j_0 - 1 - i$ в режиме gated mode. Режимы работы коррелятора устанавливаются в окне его свойств. В нем также задается размер буфера, который использует коррелятор. Параметр N задает ширину окна коррелятора. Индексы i, k, j, j_0 — целочисленные, j_0 — номер максимального шага.

6.5.8. Действительное и комплексное БПФ

Пакет VisSim/Comm содержит эффективные средства для проведения действительного и комплексного БПФ. Блок построения спектра мощности Power Spectrum обеспечивает построение спектра мощности как для действительного, так и для комплексного сигналов, что прекрасно иллюстрирует пример, показанный на рис. 6.71. При построении спектра мощности действительного сигнала на вход блока Im надо задать 0.

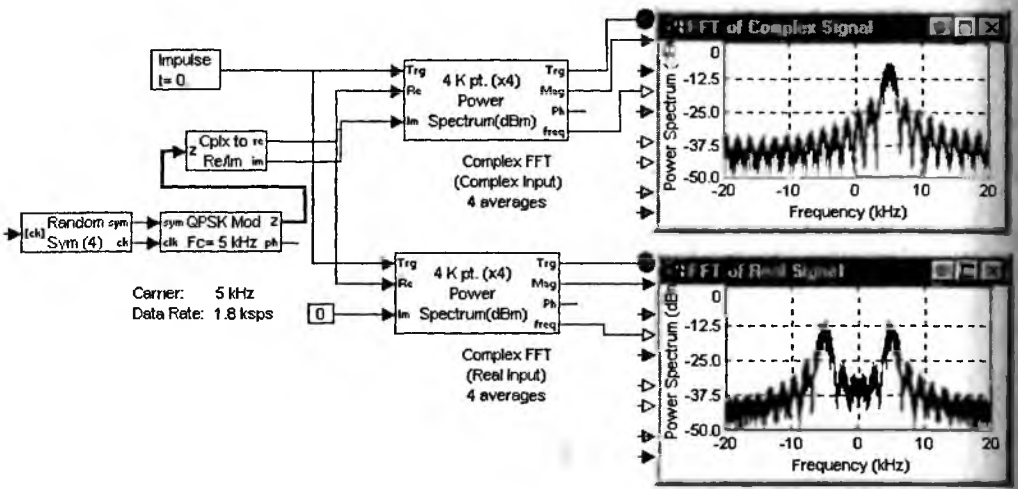


Рис. 6.71. Получение спектрограмм действительного и комплексного сигналов

Блок Power Spectrum имеет три входа — синхронизации Trg, действительной Re и мнимой Im частей входного сигнала. Блок Cplx to Re/Im обычно используется для разделения комплексного сигнала на действительную и мнимую составляющие. Различия в спектрах комплексного сигнала (осциллограмма сверху) и действительного сигнала (осциллограмма снизу) вполне очевидны.

С помощью блоков пакета VisSim/Comm можно создать анализатор спектра (рис. 6.72), весьма напоминающий такие реальные приборы. Анализатор ведет обработку и дает индикацию спектра по ходу моделирования. Спектр, показанный на рис. 6.82 справа, иллюстрирует окончательный спектр сигнала, полученный в конце моделирования. В анализаторе спектра используется блок Power Spectral Density, оценивающий мощность спектральной плотности сигнала. Спектрограммы, полученные с помощью этого блока, отличаются повышенной разрешающей способностью.

Для выполнения комплексного БПФ служит блок Complex FFT. Его применение для построения спектрограммы импульсной последовательности представлено на рис. 6.73. Здесь строится спектрограмма цифровой последовательности и приводятся данные о ее преобразовании при прямом и обратном комплексном БПФ (блоки Complex FFT и Complex IFFT).

Ниже будут представлены дополнительные примеры построения спектрограмм сигналов для ряда моделей конкретных устройств.

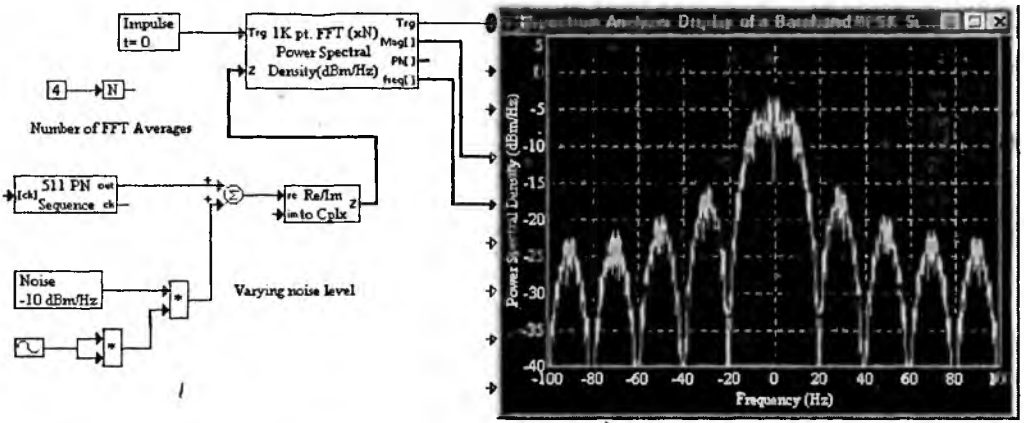


Рис. 6.72. Построение виртуального анализатора спектра

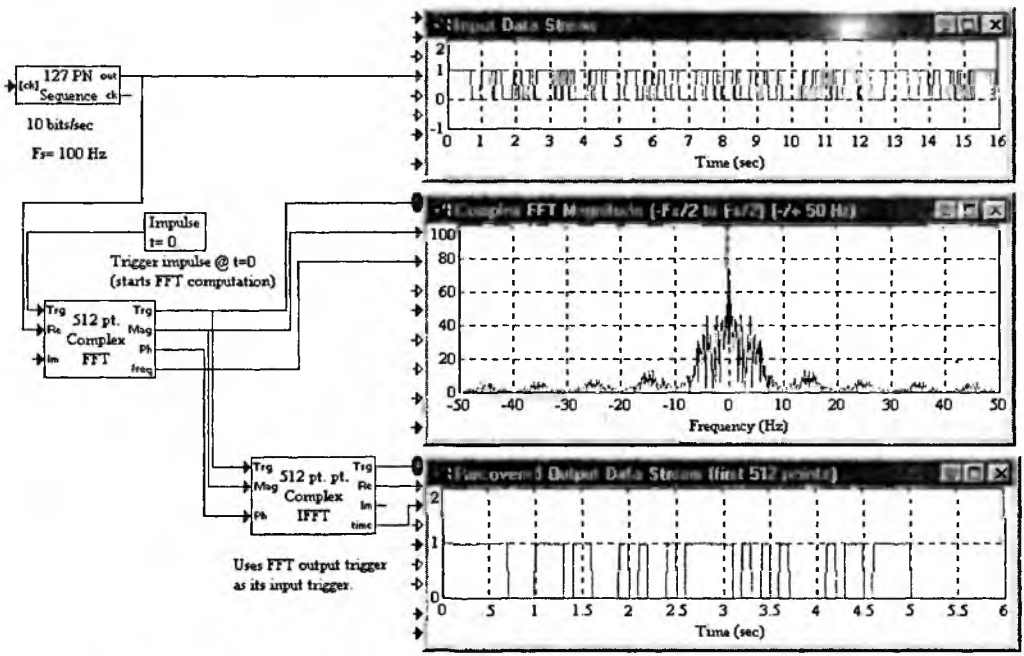


Рис. 6.73. Применение комплексного БПФ для цифровой последовательности

6.5.9. Построение и анализ цифровых фильтров

В пакете VisSim/Comm содержатся дополнительные расширенные средства для построения и анализа фильтров. Прежде всего отметим новые средства для построения IIR и FIR-фильтров.

Рисунок 6.74 поясняет задание IIR-фильтра с бесконечной импульсной характеристикой и демонстрирует реакцию такого фильтра Чебышева на синусоидальный сигнал.

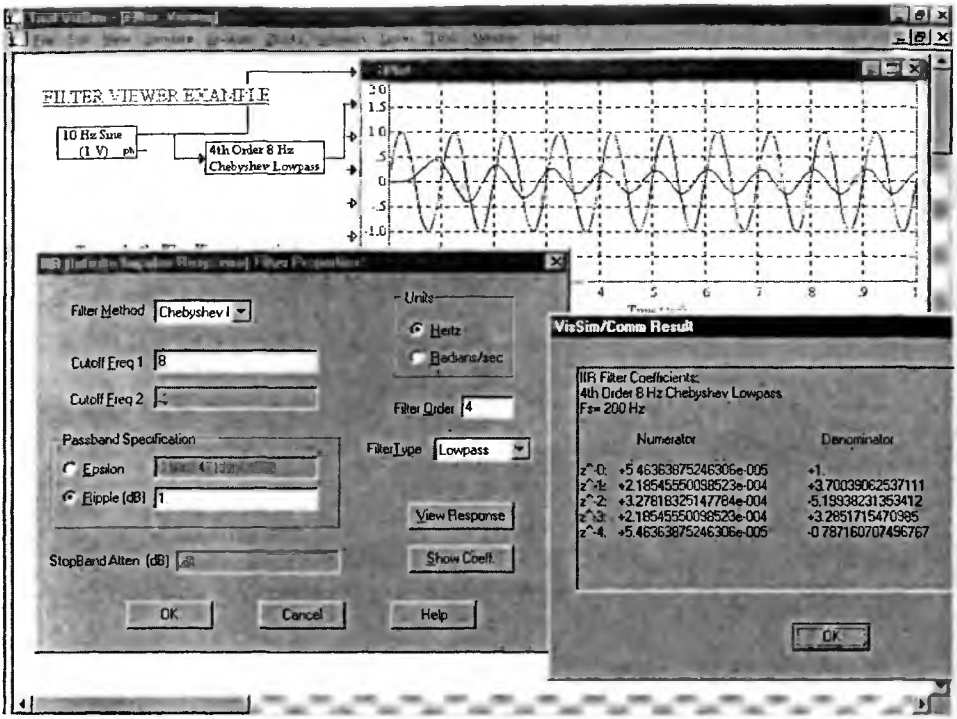


Рис. 6.74. Создание IIR-фильтра Чебышева и оценка его реакции на синусоидальный сигнал

Можно заметить, что после некоторого выброса устанавливается стационарный процесс на выходе фильтра. На рис. 6.74 представлено также окно свойств IIR-фильтра. В основном это окно повторяет окно свойств IIR-фильтров, создаваемых в VisSim, поэтому описывать повторно параметры фильтра нет смысла. Отметим лишь, что окно позволяет создавать пять типов фильтров с бесконечной импульсной характеристикой:

1) Butterworth — фильтр Баттерворта;
2) Chebyshev I — фильтр Чебышева I;
3) Chebyshev II — фильтр Чебышева II;
4) Bessel — фильтр Бесселя;
5) Elliptic — эллиптический фильтр.

Отсюда можно сделать вывод, что набор фильтров расширен. Кнопка окна свойств IIR-фильтров Show Coef. (Показать коэффициенты) выводит таблицу коэффициентов передаточной характеристки заданного фильтра. Она показана на рис. 6.84 справа от окна свойств фильтра. А кнопка View Responce выводит окно реакции фильтра (см. рис. 6.75).

В левом верхнем углу этого окна имеется набор опций, задающих вид реакции фильтра:

- Impulse — импульсная характеристика;
- Gain — амплитудно-частотная характеристика;
- Gain (dB) — амплитудно-частотная характеристика в децибелах (логарифмическая);
- Phase — фазо-частотная характеристика;

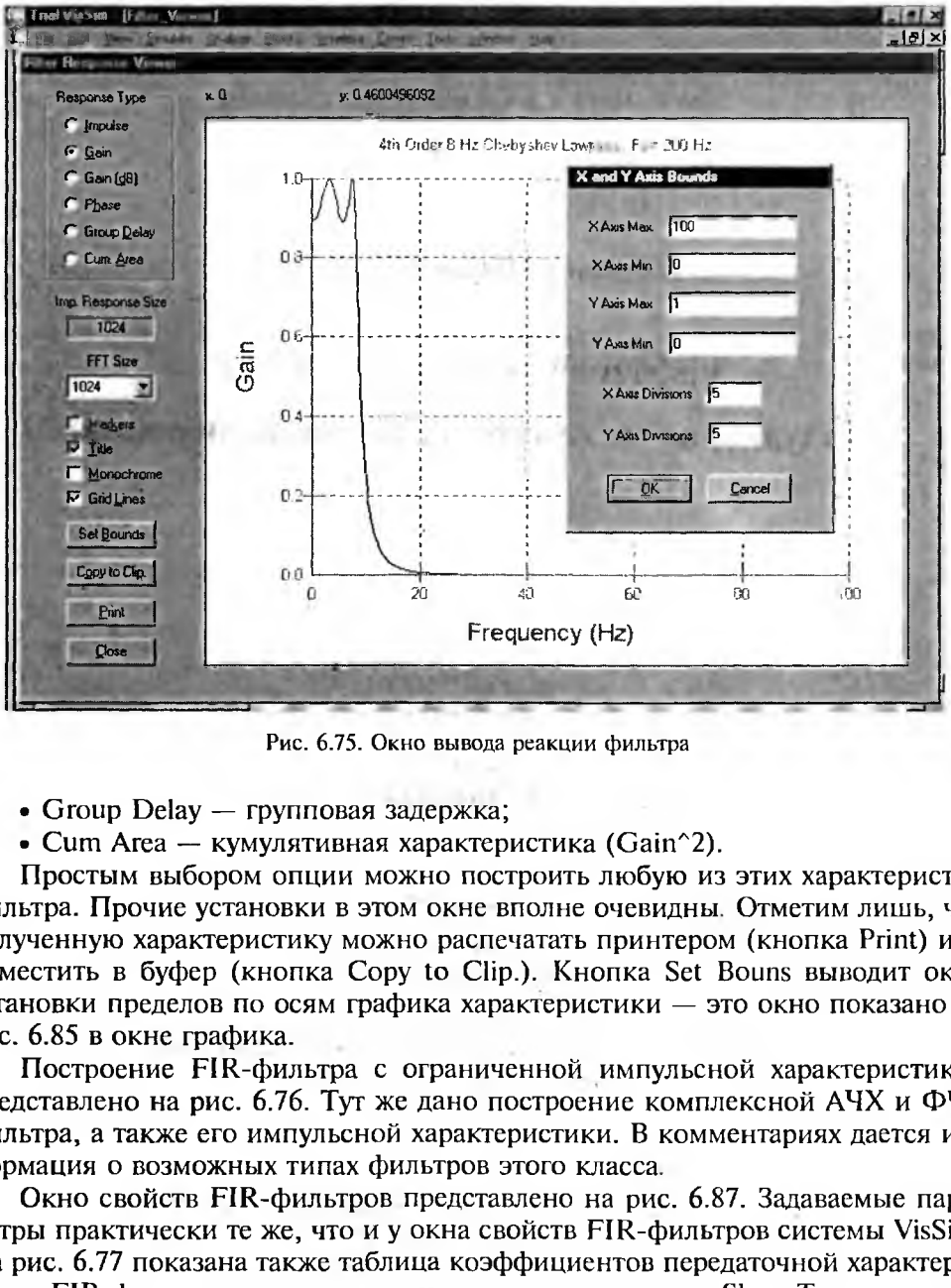


Рис. 6.75. Окно вывода реакции фильтра

- Group Delay — групповая задержка;
- Cum Area — кумулятивная характеристика (Gain^2).

Простым выбором опции можно построить любую из этих характеристи фильтра. Прочие установки в этом окне вполне очевидны. Отметим лишь, чт полученную характеристику можно распечатать принтером (кнопка Print) ил поместить в буфер (кнопка Copy to Clip.). Кнопка Set Bouns выводит окн установки пределов по осям графика характеристики — это окно показано н рис. 6.85 в окне графика.

Построение FIR-фильтра с ограниченной импульсной характеристико представлено на рис. 6.76. Тут же дано построение комплексной АЧХ и ФЧ фильтра, а также его импульсной характеристики. В комментариях дается ин формация о возможных типах фильтров этого класса.

Окно свойств FIR-фильтров представлено на рис. 6.87. Задаваемые пара метры практически те же, что и у окна свойств FIR-фильтров системы VisSim На рис. 6.77 показана также таблица коэффициентов передаточной характери стики FIR-фильтра, выводимая при активизации кнопки Show Taps.

При активизации кнопки Show Responce можно вывести окно с реакцие фильтра. Оно показано на рис. 6.78. Назначение органов управления у этог окна то же, что и у окна реакций IIF-фильтра, описанного чуть ранее.

Техника построения АЧХ и ФЧХ фильтров представлена на рис. 6.7 Здесь для построения частотных зависимостей магнитуды и фазы использует ся блок Complex FFT. Нетрудно заметить, что АЧХ и ФЧХ имеют заметны колебания.

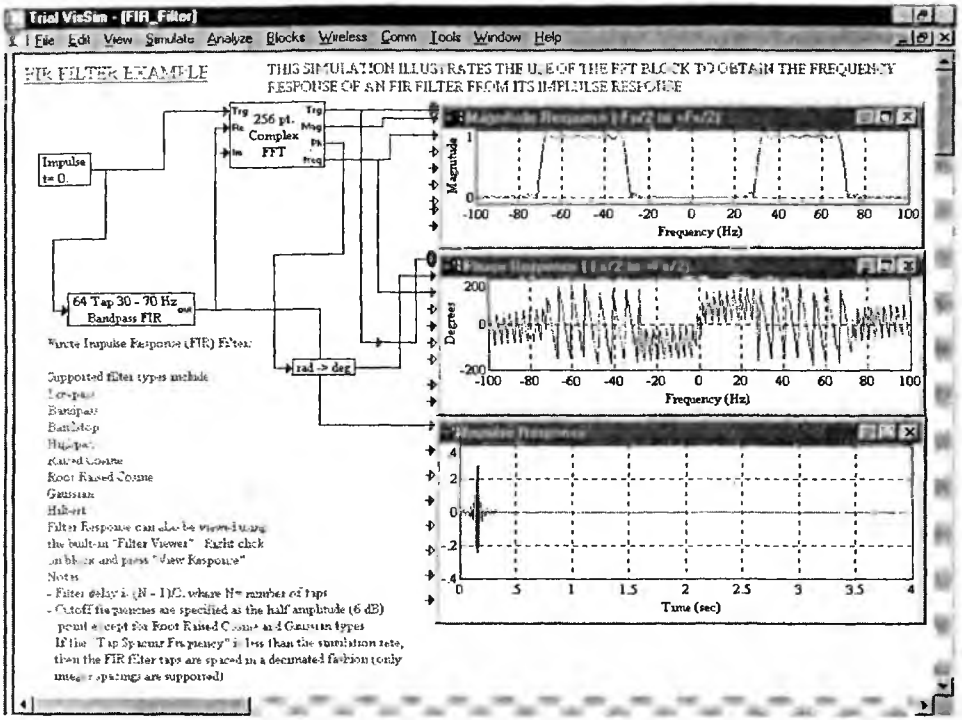
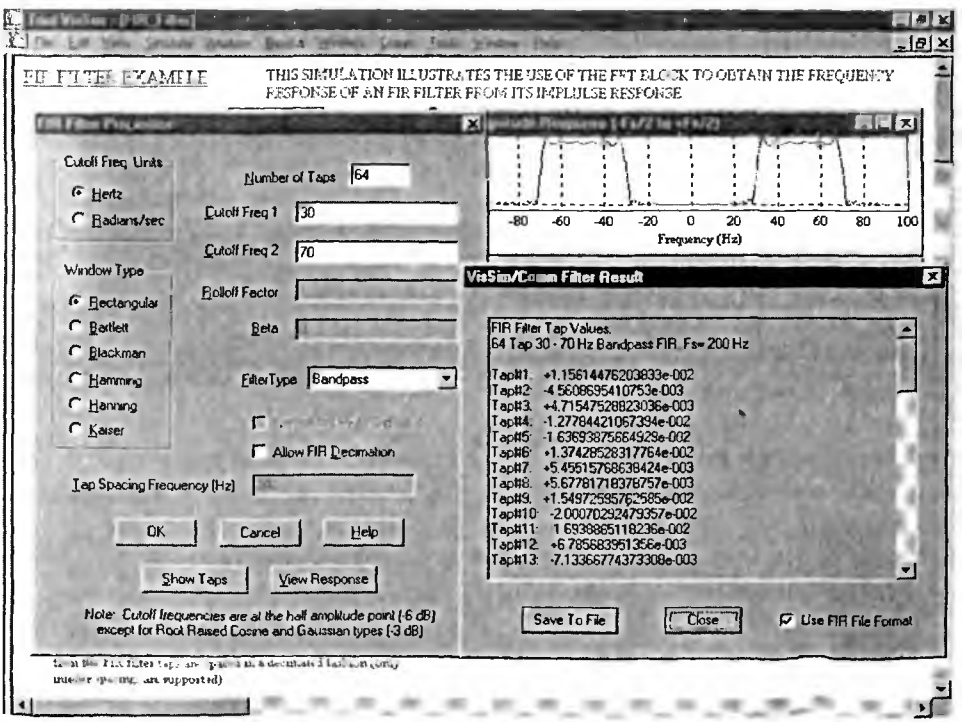


Рис. 6.76. Построение FIR-фильтра



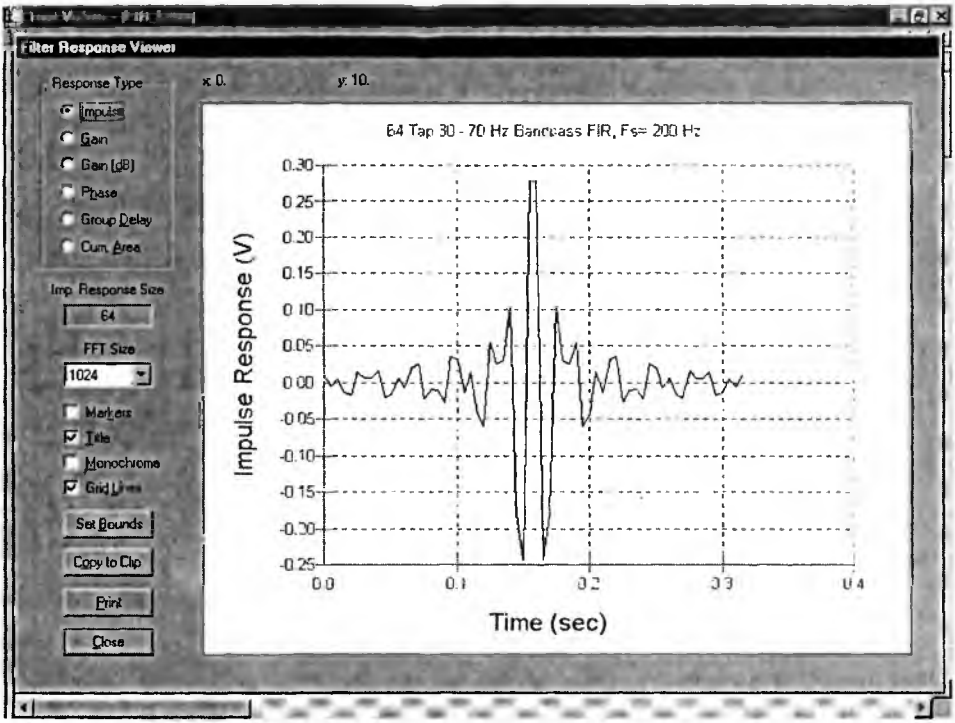


Рис. 6.78. Окно реакций FIR-фильтра с построенной импульсной характеристикой

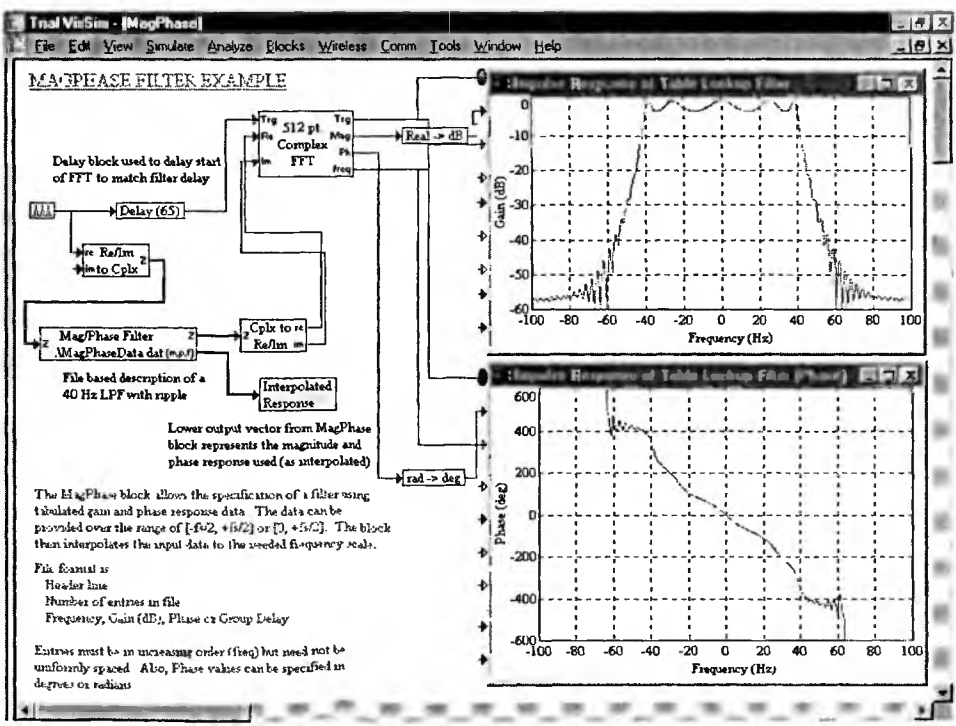


Рис. 6.79. Техника построения АЧХ и ФЧХ фильтров

На практике указанные колебания оказываются сглаженными. Для моделирования сглаживания можно применить технику интерполяции со сглаживанием. Для этого в модель рис. 6.79 встроен субблок Interpolated Responce (Интерполированная реакция). Рисунок 6.80 показывает диаграмму этого субблока построение интерполированных АЧХ и ФЧХ.

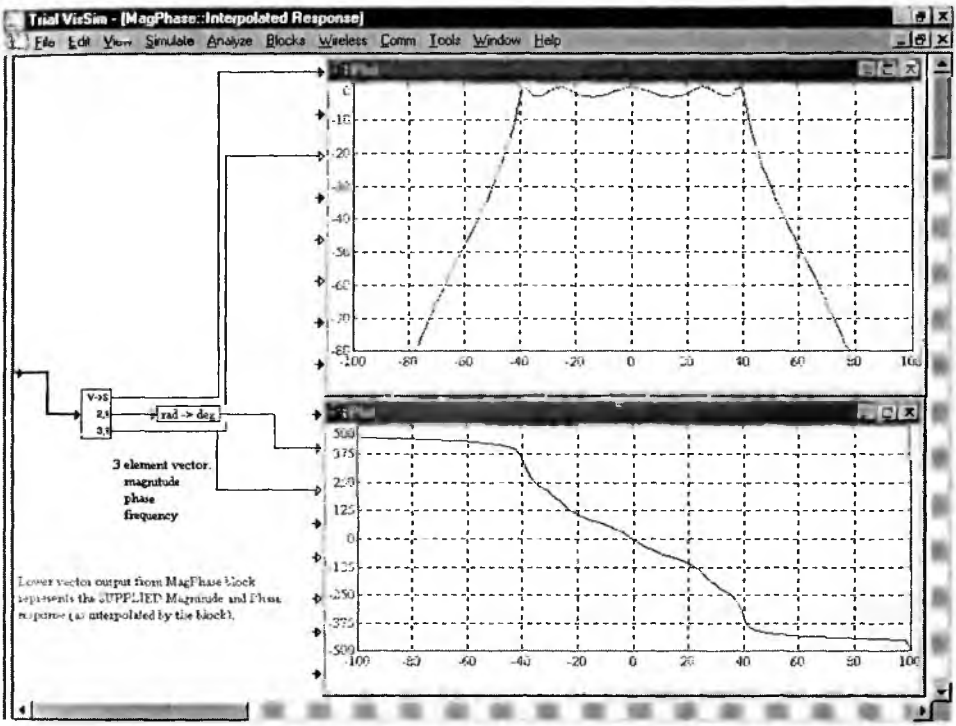


Рис. 6.80. Диаграмма субблока Interpolated Responce и АЧХ и ФЧХ после интерполяции со сглаживанием

В целом нельзя не отметить, что проектирование и анализ фильтров в системе VisSim с расширением VisSim/Comm доведены до высокой степени совершенство. Такие сложнейшие операции, как синтез и анализ IIR и FIR-фильтров, доведены до уровня простых операций, осуществляемых прямо из окон свойств фильтров. Превосходна и графическая визуализация характеристик фильтров, дающая наглядное представление о их возможностях.

6.5.10. Нелинейные устройства

Средства пакета VisSim/Comm можно использовать для построения и анализа различных нелинейных устройств. Примеры задания нелинейных устройств класса полиномов представлены на рис. 6.81.

К числу типичных нелинейных устройств относятся компрессоры (экспандеры) и декомпрессоры (экспандеры или расширители) аналоговых сигналов. Они используются для сокращения динамического диапазона таких сигналов. Применение блоков компрессии μ-Low Compress и μ-Low Expand показано на рис. 6.82.

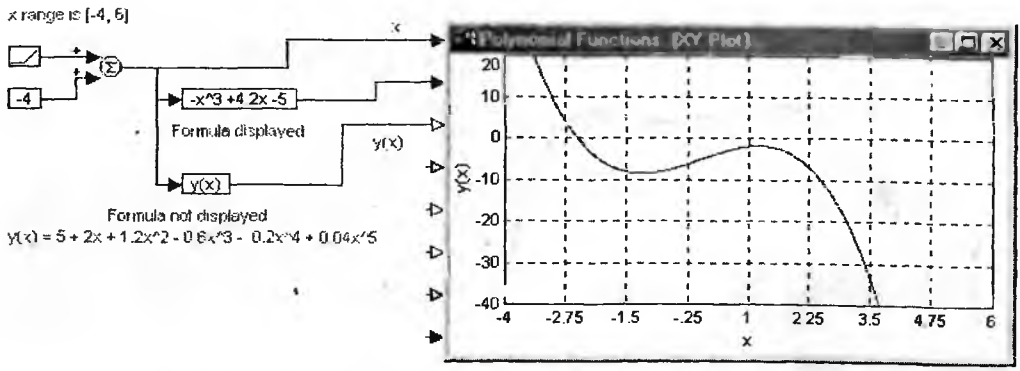


Рис. 6.81. Примеры задания нелинейностей полиномиального типа

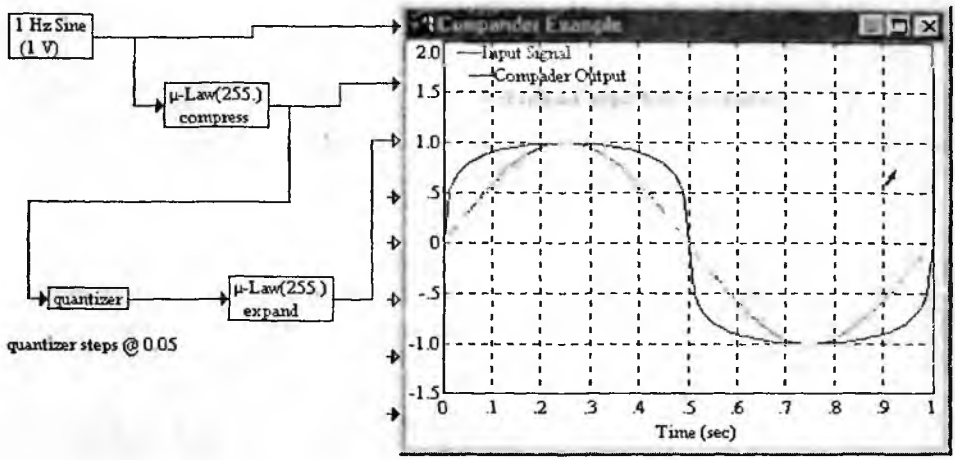


Рис. 6.82. Применение блоков μ-Low Compress и μ-Low Expand

У нелинейных устройств существует ряд типов нелинейных искажений. Искажение формы сигнала — простейший и общеизвестный тип нелинейных искажений. Он ведет к появлению только высших гармоник искаженного синусоидального сигнала. Более коварными являются интермодуляционные искажения, возникающие при подаче на нелинейные устройства двух и более синусоидальных сигналов с разными частотами. При этом могут возникнуть новые спектральные составляющие с частотами более низкими, чем у исходных сигналов. Такие искажения в звуковом диапазоне частот особенно нежелательны, поскольку новые спектральные составляющие попадают в область частот наилучшей слышимости.

Модель, иллюстрирующая возникновение интермодуляционных искажений в нелинейном усилителе, представлена на рис. 6.83. Здесь на нелинейную систему поданы синусоидальные сигналы с частотами 2 и 2,5 кГц. Блоки RF Combiner и Amplifier имитируют усилитель-ограничитель, выходной сигнал которого представлен на верхней осциллограмме, а нижняя осциллограмма представляет полученную в окне осциллографа спектрограмму сигнала. Из

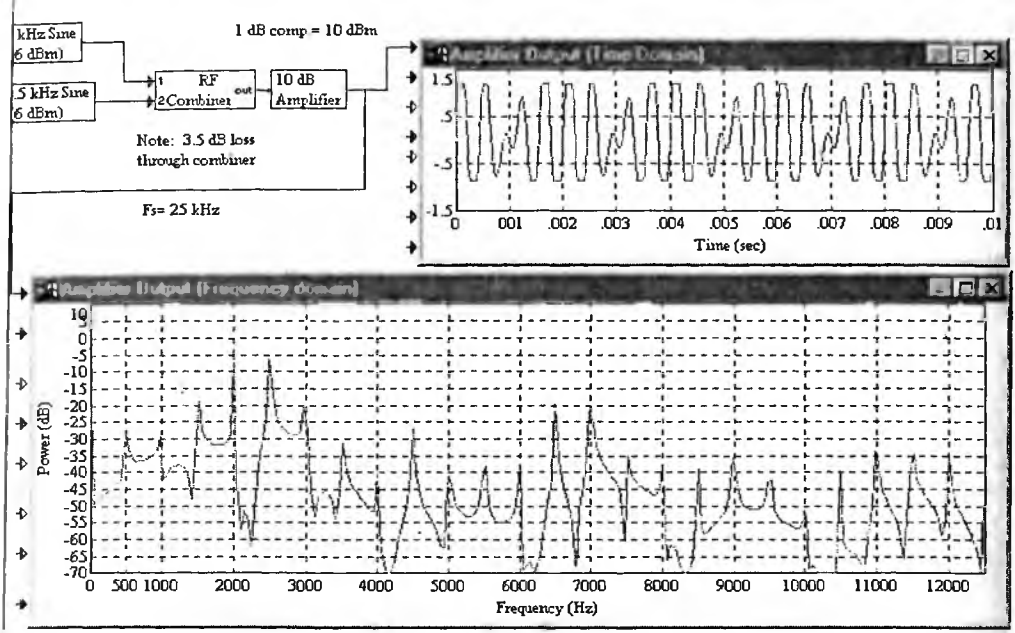


Рис. 6.83. Модель, иллюстрирующая возникновение интермодуляционных искажений в нелинейном усилителе

ıее отчетливо видны пики спектральных составляющих на частотах выше кГц, которые и являются продуктами интермодуляционных искажений.

Для получения более качественной спектрограммы можно воспользовать я той же моделью включив в нее блок Re FFT Power Spectrum (см. рис. 6.84)

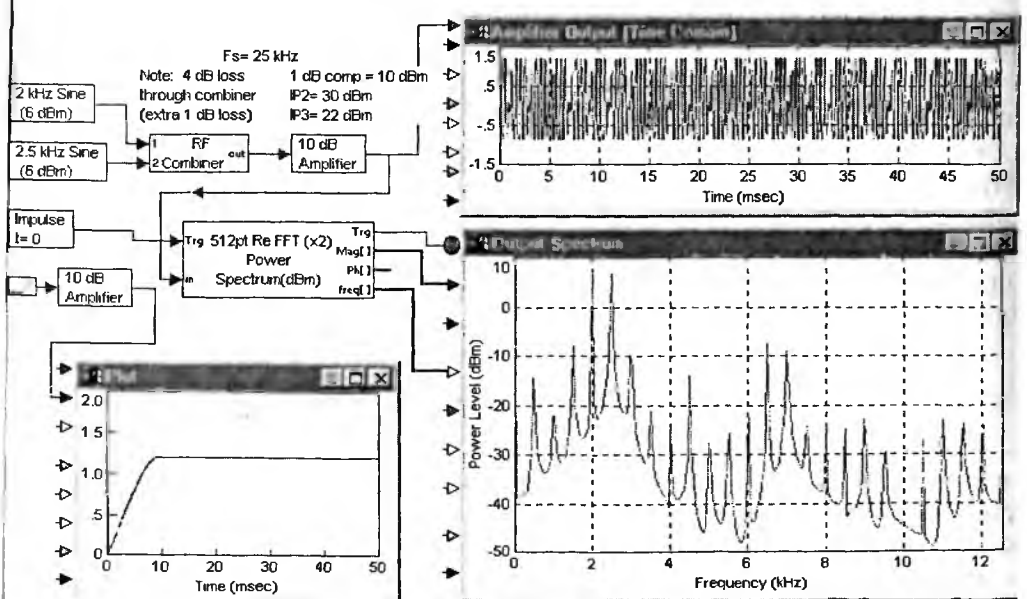


Рис. 6.84. Еще одна модель нелинейного усилителя

Теперь на спектрограмме отчетливо видны спектральные составляющие с частотами, кратными разностной частоте (500 Гц) подаваемых на вход сигналов.

Другие устройства также способны создавать нелинейные и интермодуляционные искажения. На рис. 6.85 показана модель микшера (смесителя двух сигналов) на основе блока DB Mixer, у которого задан порог ограничения. На микшер подается два звуковых сигнала с частотами 12, 5 и 8 кГц. Спектрограмма, полученная с помощью блока Re FFT Power Spectrum, отчетливо фиксирует и здесь возникновение интермодуляционных искажений, в частности появление пика разностной частоты в 4,5 кГц и гармоник этой частоты.

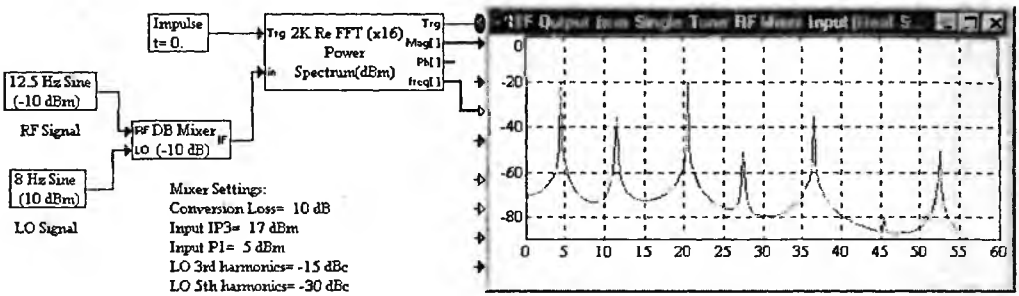


Рис. 6.85. Модель, демонстрирующая возникновение интермодуляционных искажений в микшере

6.5.11. Блок децимации

В папках Digital, Operators, PLL и RF содержатся модели ряда цифровых и иных устройств, применяемых на практике. Ввиду их многообразия остановимся только на нескольких примерах построения и анализа таких устройств.

Рисунок 6.86 иллюстрирует технику прореживания отсчетов сигналов — так называемую децимацию. Здесь создается сложный сигнал в виде суммы из 4 синусоидальных сигналов с разной амплитудой и частотой. Верхняя осциллограмма строит временную зависимость сигнала при большом числе его отсчетов (определяется числом шагов моделирования). Блок Subsample (10) обеспечивает выделение одного отсчета из 10. Две нижние осциллограммы демонстрируют получение прореженных сигналов в виде квантованных сигналов и коротких вырезок.

6.5.12. Управляемый фазовращатель

Иногда в технике связи используются так называемые фазовращатели. Так именуют устройства, позволяющие изменять сдвиг фазы. Если возможно произвольное изменение фазы под действием управляющего сигнала, то можно говорить об управляемом фазовращателе. Технику построения таких устройств иллюстрирует рис. 6.87.

Рисунок 6.87 также показывает, что вращение фазы для 4 чисел, представляющих координаты 4 точек на плоскости.

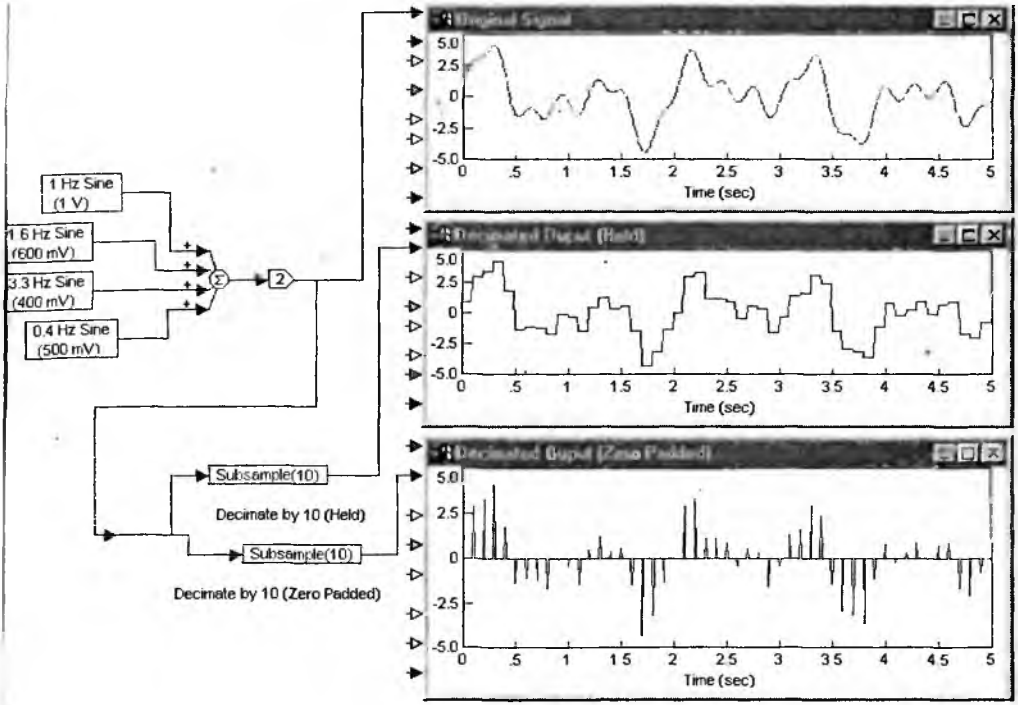


Рис. 6.86. Пример выполнения операции децимации

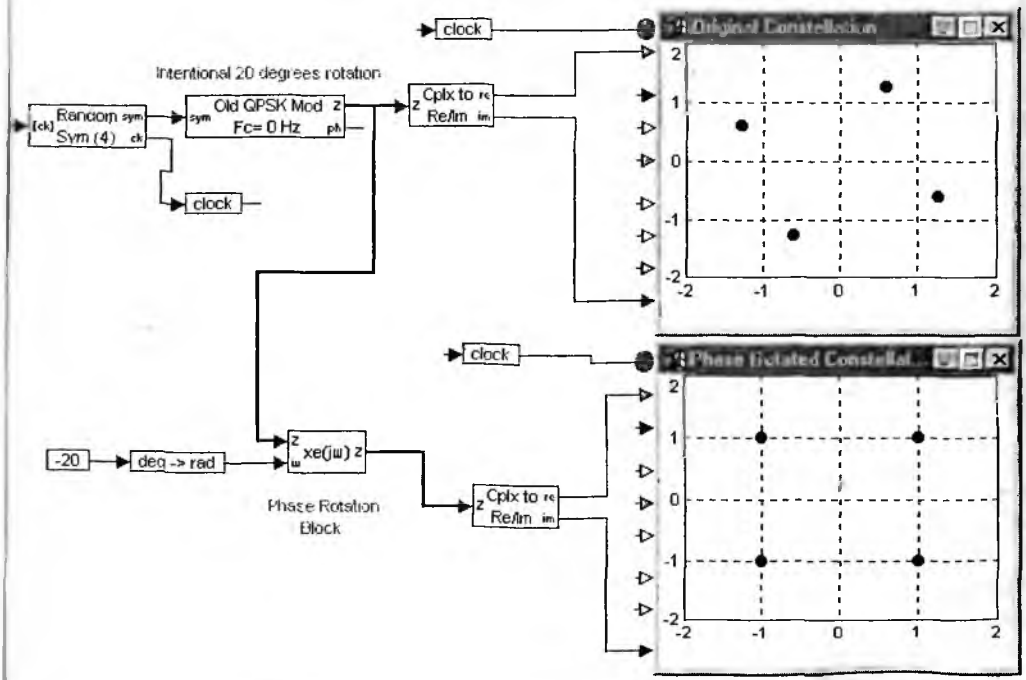


Рис. 6.87. Модель фазовращателя

6.5.13. JK-триггер

К числу весьма распространенных устройств относятся JK-триггеры — устройства с двумя состояниями равновесия, имеющие особое логическое управление. Ввиду широкой известности этих устройств мы не будем описывать их работу, а просто приведем два варианта моделей такого устройства, показанные на рис. 6.88. Сам триггер строится на основе блока Rising JK Flip Flop.

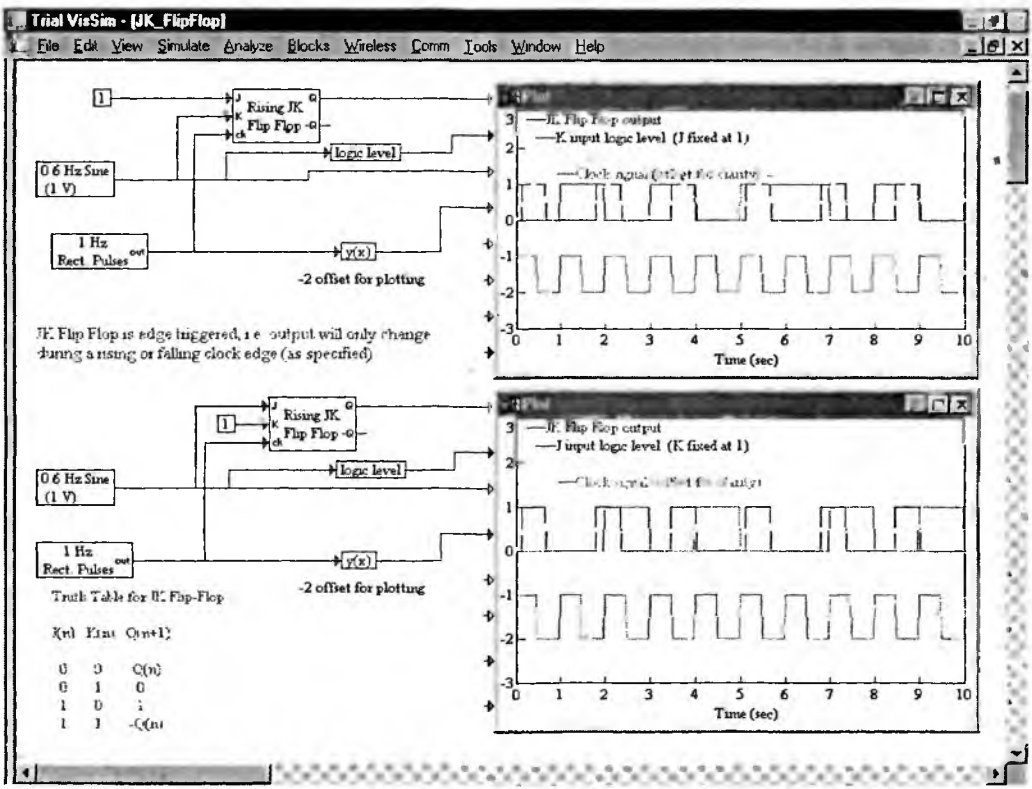


Рис. 6.88. Модель JK-триггера

6.5.14. Эквалайзер

В звуковой технике часто используются эквалайзеры — устройства, позволяющие синтезировать АЧХ из ряда частотных полос. Это позволяет выделять или наоборот подавлять отдельные области частот спектра сложных сигналов и компенсировать недостатки акустических устройств, например микрофона или громкоговорителя. Как известно, АЧХ этих устройств часто имеет довольно резкие выбросы и провалы в достаточно узких диапазонах частот.

Рисунок 6.89 показывает модель цифрового эквалайзера на основе блока Equalizer. Осциллограммы на рис. 6.89 иллюстрируют работу эквалайзера. Две

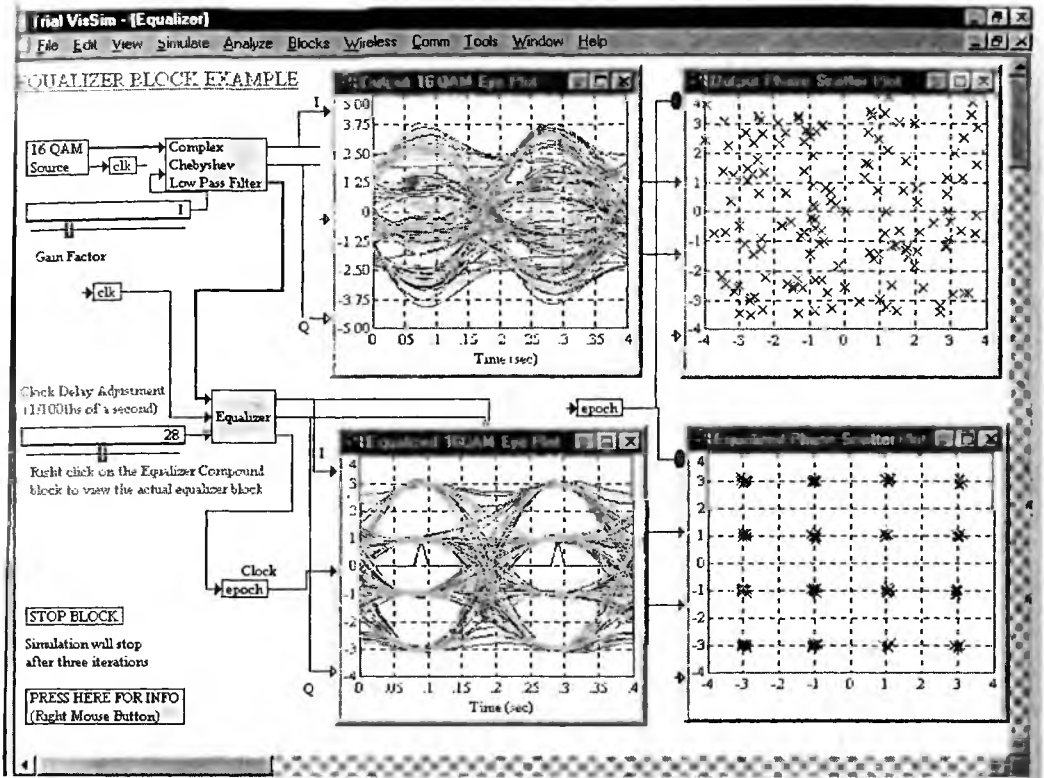


Рис. 6.89. Модель эквалайзера

осциллограммы слева показывают множество вариантов АЧХ эквалайзера, а осциллограммы справа представляют фазовые портреты устройства.

На рис. 6.89 использована специальная графика типа Scatter Plot, позволяющая наглядно отображать особые точки фазовых портретов.

6.5.15. Буферизация и обратная ей операция

В папке Digfital можно найти реализации буферизации (буфер FIFO) и обратной ей операции. Рисунок 6.90 показывает работу блоков Buffer FIFO и Unbuffer LIFO. Поскольку в главе 3 мы подробно обсуждали работу буфера, то работа этих блоков должна быть понятной. Стоит лишь отметить, что в этом примере моделирование выполняется в реальном времени.

Для преобразования символов в биты и наоборот служат блоки Sym->N и N->Sym, где целое число N — разрядность преобразования (в битах). Они могут работать в режимах (модах) LSB и MSB. Рисунки 6.91 и 6.92 поясняют применение этих блоков. N не может быть декларировано как глобальная переменная. Окна свойств этих блоков показаны на этих рисунках. В них предусмотрено задание N и режима преобразования.

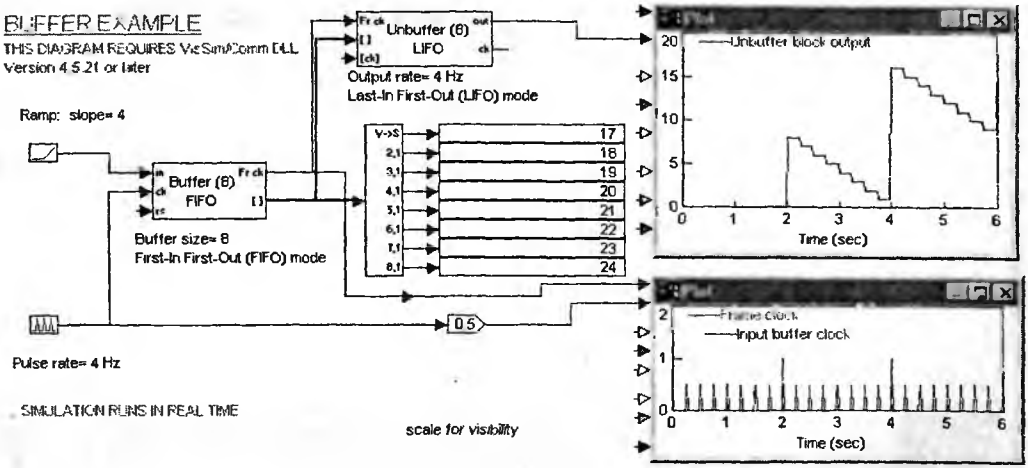


Рис. 6.90. Применение блоков Buffer FIFO и Unbuffer LIFO

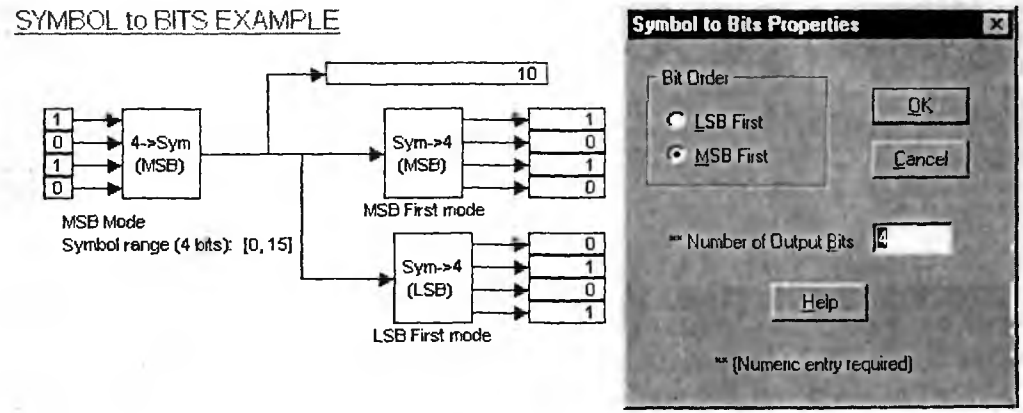


Рис. 6.91. Преобразование символов в биты

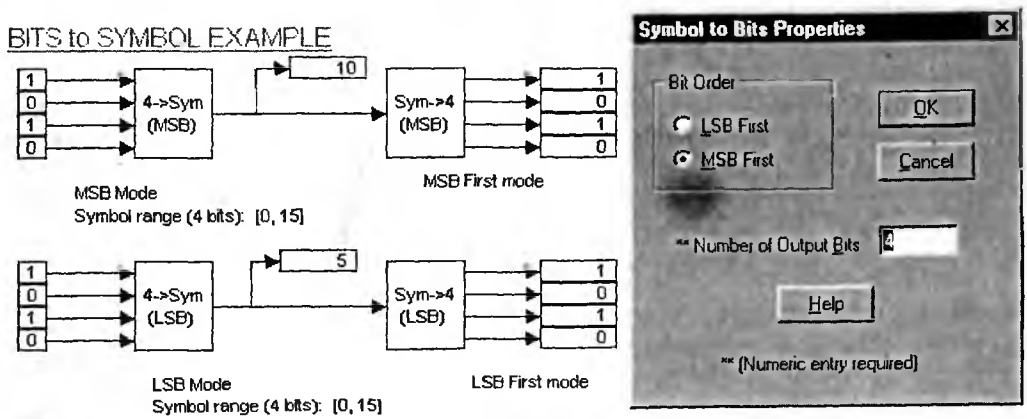


Рис. 6.92. Преобразование битов в символы

6.5.16. Делители частоты повторения импульсов

Для построения *делителей частоты* повторения импульсных сигналов лужит блок Div by N, где целое число N — коэффициент деления. Рисунок .93 дает примеры применения блока Div by N для разных N и для разных становок в окне свойств этого блока, которое также показано на рис. 6.93. окне свойств можно установить коэффициент деления, временной сдвиг и ороговый уровень срабатывания блока. Кроме того, можно задать выход в иде коротких импульсов или импульсов прямоугольной формы с равными лительностями вершины и полки импульсов.

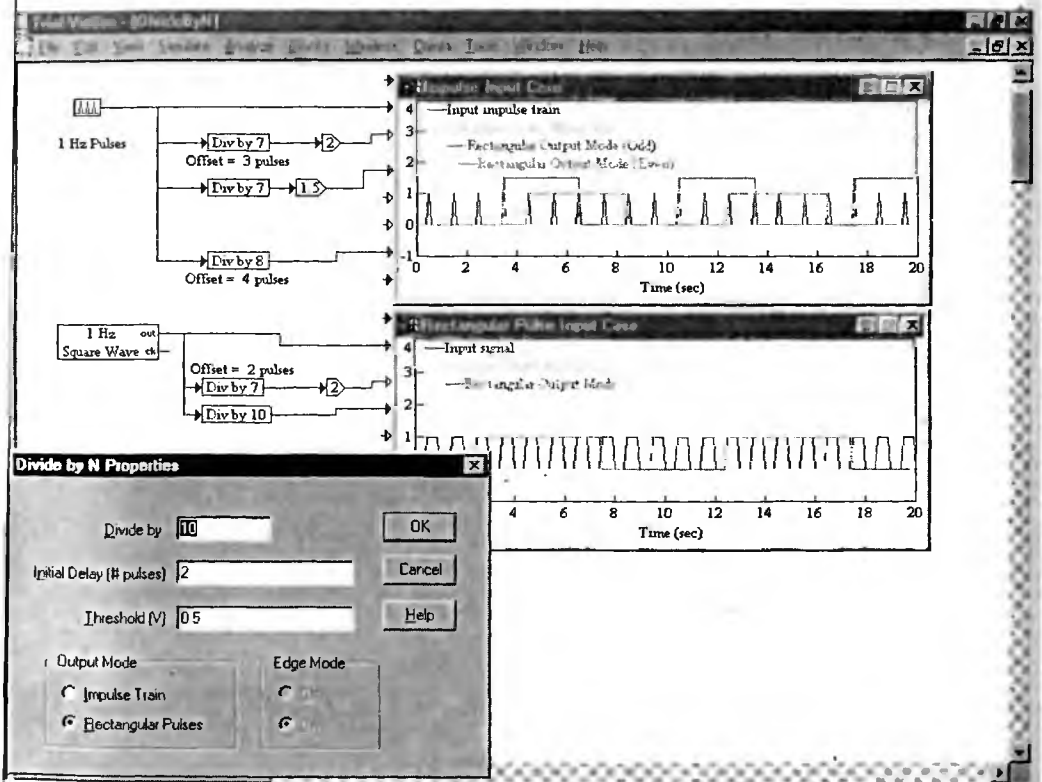


Рис. 6.93. Примеры применения блока Div by N

6.5.17. Организация очередей и стеков

К числу важнейших классических типов данных относятся очереди. Блок Queue служит для организации очереди. Его применение представлено на рис. 6.94. Блок может использовать типы очередей FIFO (первый пришел, первый вышел) и LIFO (последний пришел, первый вышел). Очереди типа LIFO известны так же как стеки.

На рис. 6.94 снизу представлено окно свойств блока Queue. Оно позволяет задать тип очереди и число мест в ней, а также выход при пустой очереди.

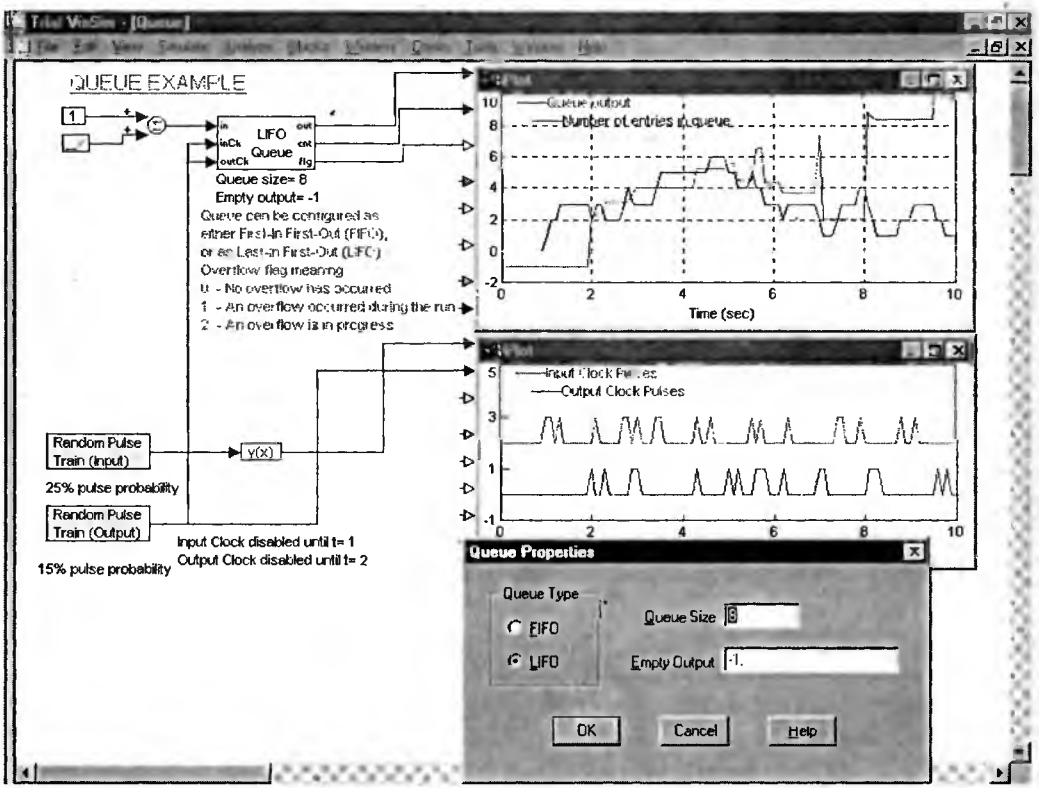


Рис. 6.94. Организация очереди с помощью блока Queue

6.5.18. Преобразователи способов представления информации

К основным способам представления и передачи информации по коммуникационным каналам относятся параллельный и последовательный способы. Для демонстрации преобразований этих видов в папке Digital имеются соответствующие демонстрационные примеры. Рисунок 6.95 демонстрирует преобразование параллельного способа передачи информации в последовательный способ. Для этого предназначен блок Parallel (N) to Serial.

На другом рисунке (рис. 6.96) показано преобразование последовательного способа представления информации в параллельный. Для этого используется блок Serial to Parallel.

В окнах настройки этих блоков можно установить порядок использования битов LSB или MSB.

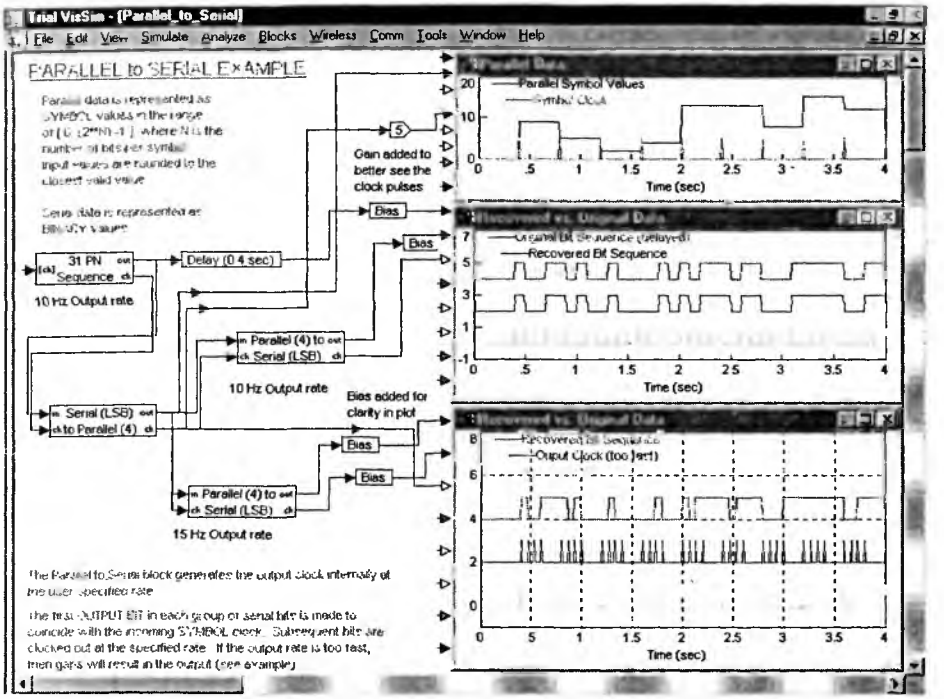


Рис. 6.95. Преобразование параллельного способа представления информации в последовательный

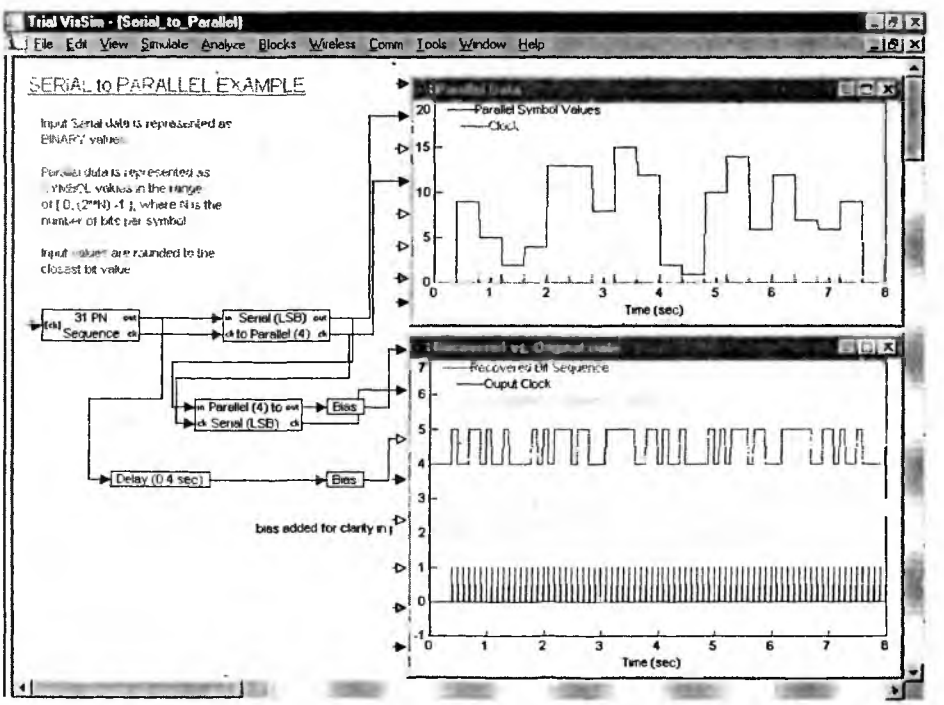


Рис. 6.96. Преобразование последовательного способа представления информации в параллельный

6.5.19. Моделирование системы с дифференциальной фазовой модуляцией

В решении ряда задач коммуникационной техники широкое применение находят системы модуляции, основанные на изменении фазового сдвига синусоидальных или импульсных сигналов. Модуляторы и демодуляторы для этого вида модуляции относятся к типу PSK (Phase Shift Keying). В папках демонстрационных примеров Modulators и Demodulators пакета VisSim/Com можно найти примеры на осуществление нескольких видов такой модуляции. Ограничимся примером дифференциальной PSK. Для этого вида модуляции имеются блоки DBPSK, DQPSK, /4- DQPSK, D8PSK, D16PSK и D32PSK. Возможна реализация для действительного и комплексного представлений.

Рисунок 6.97 показывает реализацию модели с блоком DQPSK. С помощью генератора случайных чисел имитируется поток кодов текстовых символов. Для обеспечения надежности фазовой модуляции используются специальные переходы от одного сдвига фазы к другому с помощью специальной карты переходов. Она для идеального случая показана на комплексной плоскости на верхней осциллограмме, а в реальном случае соответствует множеству точек, построенных диагональными крестиками (вторая осциллограмма сверху).

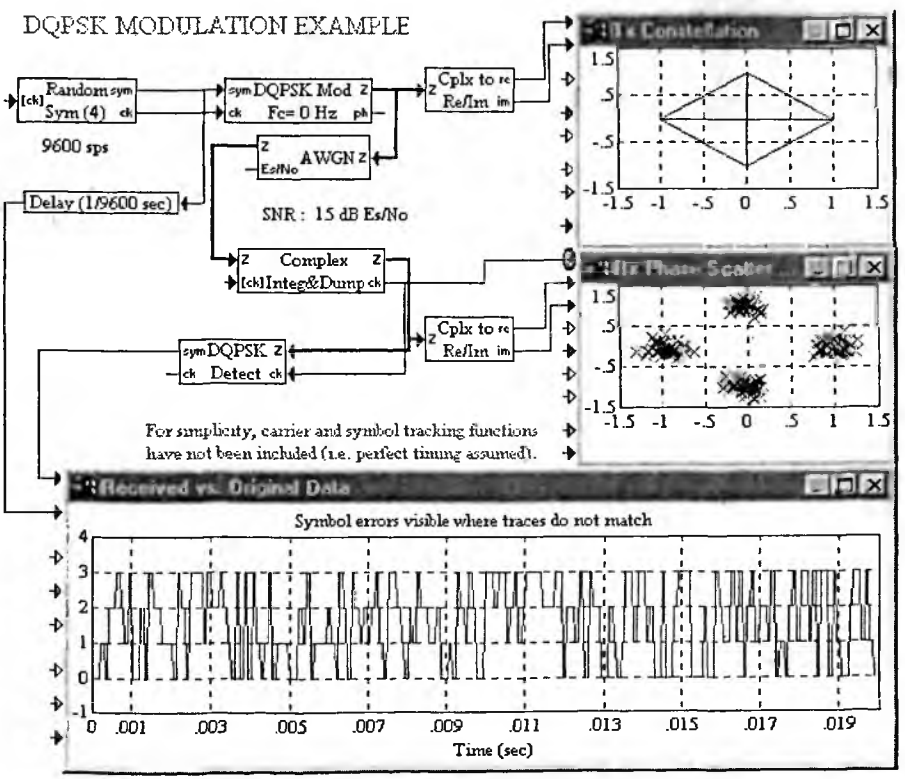


Рис. 6.97. Имитация передачи текстовых кодов с применением дифференциальной фазовой модуляции

После демодуляции блоком DQPSK Detect получается выходной поток кодов, практически повторяющий входной поток. В результате оба графика (в оригинале красного и синего цветов) сливаются, что свидетельствует об отсутствии ошибки в передаче текстовых кодов. Максимальной число кодов ограничено значением 100. Окно свойств блока DQPSK Mod представлено на рис. 6.98.

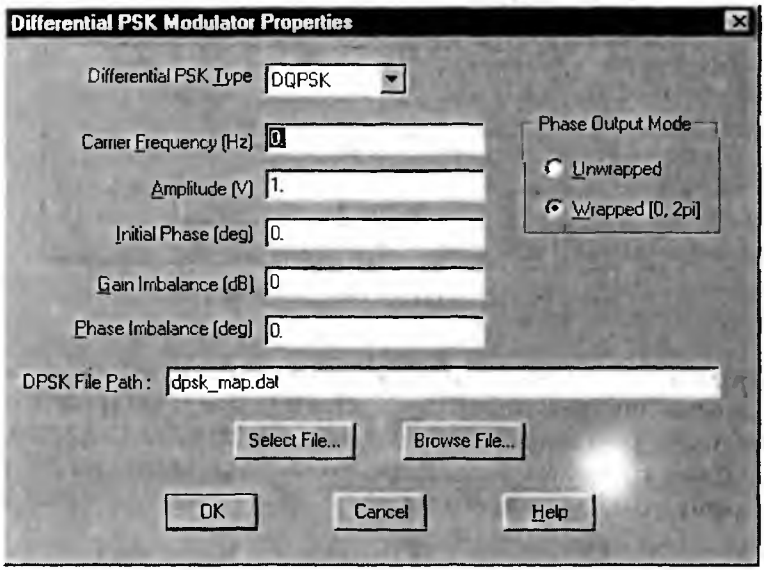


Рис. 6.98. Окно свойств блока DQPSK Mod

В этом окне имеются следующие установки:

- DPSK Type — задает тип модуляции (DBPSK, DQPSK, /4- DQPSK, D8PSK, D16PSK или D32PSK);
- Carrier Frequency — задает частоту несущей в герцах (установите 0 при комплексном представлении);
- Amplitude — задает амплитуду в вольтах;
- Initial Phase — задает начальную фазу в градусах;
- Gain Imbalance — разбаланс по коэффициентам передачи (Q относительно I) в децибелах;
- Phase Imbalance — разбаланс по фазам (Q относительно I);
- Phase Output Mode — задает характер изменения фазы (без границ — Unwrapped и в заданных пределах [0, 2π] — Wrapped);
- Select File — открывает окно для выбора файла, задающего карту фазовых изменений;
- Browse File — открывает окно текстового редактора Notepad для просмотра файла с картой фазовых изменений;
- DPSK File Path — устанавливает (по правилам MS-DOS) путь к файлу фазовых изменений.

Окно свойств блока DQPSK Detect показано на рис. 6.99. Его установочные параметры были описаны выше.

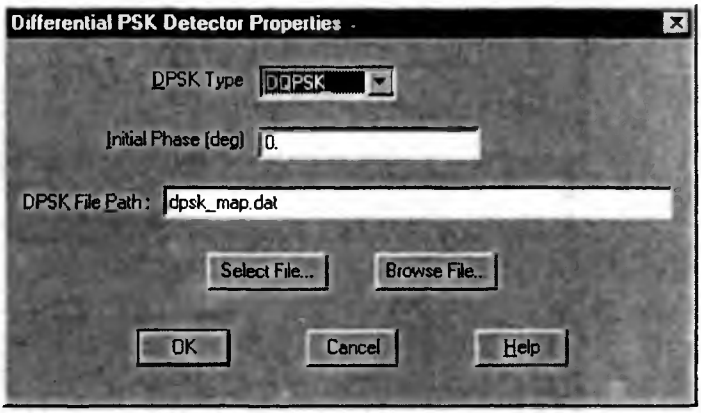


Рис. 6.99. Окно свойств блока DQPSK Detect

В целом можно сделать заключение, что применение системы VisSim с дополнительными программными продуктами резко расширяет возможности данной системы и превращает ее в мощный инструмент визуально-ориентированного математического моделирования огромного числа систем и устройств самого разного назначения.

Список литературы

1. *Самарский А. А., Михайлов А. П.* Математическое моделирование: Идеи. Методы. Примеры. 2. изд., испр. М.: Физматлит, 2001.

3. *Бенькович Е. С., Колесов Ю. Б., Сениченков Ю. Б.* Практическое моделирование динамических систем. СПб.: БХВ-Петербург, 2002.

4. *Максимей И. В.* Имитационное моделирование на ЭВМ. М.: Радио и связь, 1988.

5. *Шеннон Р.* Имитационное моделирование систем. Искусство и наука. М.: Мир, 1978.

6. Семененко М. Г. Введение в математическое моделирование. М.: Солон-Р, 2002.

7. Математическое моделирование. / Под ред. Дж. Эндрюса, Р. Мак-Лоуна; Пер. с англ. М.: Мир, 1979.

8. Введение в математическое моделирование: Учебное пособие / В. Н. Ашихмин и др.; Под ред. П. В. Трусова. М.: Интермет Инжиниринг, 2000.

9. *Цисарь И. Ф., Нейман В. Г.* Компьютерное моделирование экономики. М.: Диалог-МИФИ, 2002.

10. *Шелобаев С. И.* Математические методы и модели в экономике, финансах, бизнесе: Учеб. пособие для вузов. М.: ЮНИТИ-ДАНА, 2001.

11. 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.

12. *Прицкер А.* Введение в имитационное моделирование и язык СЛАМ II. М.: Мир, 1987.

13. *Дьяконов В. П.* Расчет нелинейных и импульсных устройств на программируемых микрокалькуляторах: Справочное пособие. М.: Радио и связь, 1984.

14. *Дьяконов В. П.* Справочник по расчетам на микрокалькуляторах. М.: Наука, Физматлит, 1989.

15. *Трохименко Я. К., Любич Ф. Д.* Радиотехнические расчеты на микрокалькуляторах. М.: Радио и связь, 1983.

16. *Дьяконов В. П.* Применение персональных ЭВМ и программирование на языке Бейсик. М.: Радио и связь, 1989.

17. *Разевиг В. Д.* Система сквозного проектирования электронных устройств DesignLab 8.0. М.: Солон, 1999.

18. *Разевиг В. Д.* Система схемотехнического моделирования MicroCAP V. М.: Солон, 1997.

19. *Карлащук В. И.* Электронная лаборатория на IBM PC. М.: Солон, 1999.

20. *Дьяконов В. П.* Компьютерная математика. Теория и практика. М.: Нолидж, 2001.

21. *Дьяконов В. П.* Системы компьютерной математики Derive. Самоучитель. М.: Солон-Р, 2002.

22. *Дьяконов В. П., Абраменкова И. В.* Mathcad 8 PRO в математике, физике и Internet. М.: Нолидж, 1999.

23. *Дьяконов В. П.* MathCAD 2001: Специальный справочник. СПб.: Питер, 2002.

24. *Дьяконов В. П.* Maple 7: Учебный курс. СПб.: Питер, 2002.

25. *Дьяконов В. П.* Mathematica 4: Учебный курс. СПб.: Питер, 2001.

26. *Дьяконов В. П.* MATLAB 6: Учебный курс. СПб.: Питер, 2001.

27. *Дьяконов В. П.* MATLAB 6/6.1/6.5. Simulink 4.5. Основы применения: Полное руководство пользователя. М.: Солон-Пресс, 2002.

28. *Дьяконов В. П.* MATLAB 6/6.1/6.5. Simulink 4.5 в математике и моделировании. Полное руководство пользователя. М.: Солон-Пресс, 2003.

29. *Дьяконов В. П.* Simulink 4. Специальный справочник. СПб.: Питер, 2002.

30. *Дьяконов В. П., Круглов В. В.* Математические пакеты расширения MATLAB. Специальный справочник. СПб.: Питер, 2001.

31. *Дьяконов В. П., Круглов В. В.* MATLAB. Анализ, идентификация и моделирование систем. Специальный справочник. СПб.: Питер, 2002.

32. *Дьяконов В. П., Абраменкова И. В.* MATLAB. Обработка сигналов и изображений. СПб.: Питер, 2002.

33. *Бурсиан Э. В.* Задачи по физике для компьютера: Учеб. пособие для студентов физ.-мат. фак. пед. ин-тов. М.: Просвещение, 1991.

34. *Зернов Н. В., Карпов В. Г.* Теория радиотехнических цепей. Л.: Энергия, 1972.

35. *Бронштейн И. Н., Семендяев К. А.* Справочник по математике для инженеров и учащихся втузов М.: Наука, Физматлит, 1990.

36. Математический энциклопедический словарь / Под ред. Ю. В. Прохорова. М.: Советская энциклопедия, 1988.

37. *Дьяконов В. П.* Вейвлеты. От теории к практике. М.: Солон-Р, 2002.

38. *Шредер М.* Фракталы, хаос, степенные законы. Миниатюры из бесконечного рая. Пер. с английского. Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001.

39. *Сергиенко А. Б.* Цифровая обработка сигналов. СПб.: Питер, 2002.

40. *Топчеев Ю. И.* Атлас для проектирования систем автоматического управления: Учебное пособие для втузов. М.: Машиностроение, 1989.

Содержание

Серия «Полное руководство пользователя»

Владимир Павлович Дьяконов

VisSim+Mathcad+MATLAB
Визуальное математическое моделирование

Ответственный за выпуск
В. Митин

Макет и верстка
С. Тарасов

Обложка
Е. Холмский

ООО «СОЛОН-Пресс»
123242, г. Москва, а/я 20
Телефоны:
(095) 254-44-10, (095) 252-36-96, (095) 252-25-21
E-mail: Solon-Avtor@coba.ru

По вопросам приобретения обращаться:
ООО «Альянс-книга»
Тел: (095) 258-91-94, 258-91-95

ООО «СОЛОН-Пресс»
127051, г. Москва, М. Сухаревская пл., д. 6, стр. 1 (пом. ТАРП ЦАО)
Формат 70×100/16. Объем 24 п. л. Тираж 1000

Отпечатано в ООО "Джейнсер"


05 285786 030005 2