Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Ульяновский государственный технический университет

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ЭЛЕКТРОПРИВОДЕ

Ульяновск 2006

Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Ульяновский государственный технический университет

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ЭЛЕКТРОПРИВОДЕ

Конспект лекций по курсу
для студентов специальности
«Электропривод и автоматика промышленных установок и комплексов»

Составитель: А. В. Доманов

Ульяновск 2006

УДК 62-83.001.63(076)
ББК 31.291 я7
К 63

Рецензенты:
кафедра аэронавигации и радиоэлектронного оборудования УВАУГА (институт) (зав. кафедрой Е. В. Антонец);
зам. главного инженера ЗАО «ФРЕСТ» канд. техн. наук М. И. Коваль.

Одобрено секцией методических пособий научно-методического совета УлГТУ.

Компьютерные технологии в электроприводе: конспект лекций / К63 сост. А. В. Доманов. – Ульяновск : УлГТУ, 2006. – 112 с.

Составлен в соответствии с программой курса «Компьютерные технологии в электроприводе» для студентов всех форм обучения. В конспекте лекций приведены описания современных пакетов программ, используемых при проектировании и эксплуатации систем электропривода. Рассмотрены пакеты: VisSim, Pspice, Simulink, MVS, ELCUT, LabView, AVR Studio; а также вопросы построения распределенных систем с использованием электропривода.

Работа подготовлена на кафедре «Электропривод и автоматизация промышленных установок».

УДК 62-83.001.63(076)
ББК 31.291 я7

Учебное издание

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ЭЛЕКТРОПРИВОДЕ

Конспект лекций

Составитель ДОМАНОВ Андрей Викторович

Редактор Л. В. Рыжова

Подписано в печать 10.03.2006. Формат 60×84/16. Бумага офсетная.

Печать трафаретная. Усл. печ. л. 7,0. Уч.-изд. л. 6,0.

Тираж 75 экз. Заказ

Ульяновский государственный технический университет

432027, Ульяновск, Сев. Венец, 32.

Типография УлГТУ, 432027, Ульяновск, Сев. Венец, 32.

ОГЛАВЛЕНИЕ

Введение

Курс «Компьютерные технологии в электроприводе» посвящен эффективному применению современных компьютерных технологий в проектировании, создании и эксплуатации электроприводов.

Компьютерные технологии в электроприводе интенсивно развиваются за счет:

- развития самого электропривода;
- развития компьютерной техники и программного обеспечения;
- все более широкого применения цифровых технологий.

Компьютерные технологии в электроприводе можно разделить на несколько групп по применению:

- анализ и синтез электропривода (моделирование);
- проектирование электронной части ЭП и печатных плат;
- программирование и отладка цифровых систем управления (МК, ПЛИС);
- построение распределенных систем, использование виртуальных панелей управления и контроля.

Рассмотрим более подробно каждую группу.

Моделирование широко и давно применяется для анализа и синтеза электропривода. Первые модели были физические, затем – на основе аналоговых элементов, сейчас им на смену пришло компьютерное моделирование. Существует большое количество моделирующих программ, их можно разделить на следующие группы:

- математического моделирования;
- моделирования структурных схем по передаточным функциям;
- моделирования электронных схем;
- моделирования физических процессов, полей и т. п.

Программы математического моделирования применяются на этапе первичного анализа и позволяют определить некоторые численные характеристики разрабатываемой системы.

Программы моделирования структурных схем позволяют оценить реакцию системы на различные управляющие и возмущающие воздействия, провести анализ переходных процессов, а также провести частотный анализ. Используя такие программы, оценивается правильность выбора структурной схемы и корректирующих звеньев.

Программы моделирования электронных схем применяются для анализа и оптимизации электронной части электропривода с учетом характеристик конкретных используемых элементов, вплоть до анализа помехозащищенности и тепловых режимов работы.

Программы моделирования полей позволяют рассчитывать электрические, магнитные и тепловые поля различных систем.

Пакеты программ для проектирования электронных схем позволяют собирать из элементов принципиальные схемы электронных узлов электропривода, проводить их оптимальное размещение в заданных габаритах печатной платы, проводить автоматическую трассировку платы, подготавливать проектную документацию для изготовления печатных плат.

Создание цифровых систем управления электроприводом также невозможно без использования компьютерных технологий. Все управляющие сигналы, а также данные с датчиков обрабатываются микроконтроллером или программируемой логикой. При этом проектировщик должен разработать и отладить управляющую программу для микроконтроллера. Хотя языки программирования различных микроконтроллеров очень похожи, каждая фирма-производитель имеет свое программное обеспечения для работы со своими микроконтроллерами. Многие из них включают в пакет программ не только компилятор для перевода программы, написанной разработчиком в исполняемый код микроконтроллера, но и средства отладки и эмуляторы микроконтроллеров.

Относительно новым блоком компьютерных технологий в электроприводе является построение распределенных систем и использование виртуальных панелей управления и контроля. Эти технологии позволяют дистанционно анализировать работу удаленных электроприводов и управлять ими с единого диспетчерского пульта.

В данном курсе будут рассмотрены наиболее распространенные и широко применяемые пакеты программ.

1. Моделирование структурных схем (VisSim)

Программа VisSim предназначена для построения, исследования и оптимизации виртуальных моделей физических и технических объектов, в том числе и систем управления. VisSim – это аббревиатура выражения Visual Simulator – визуальная, воспринимаемая зрением, среда и средство моделирования.

Программа VisSim разработана и развивается компанией Visual Solutions (USA). Эта программа – мощное, удобное в использовании, компактное и эффективное средство моделирования физических и технических объектов, систем и их элементов.

Программа предоставляет разработчику развитый графический интерфейс, используя который он создает модель из виртуальных элементов с некоторой степенью условности так же, как если бы он строил реальную систему из настоящих элементов. Это позволяет создавать, а затем исследовать и оптимизировать модели систем широкого диапазона сложности [1].

При описании и последующем построении модели в среде VisSim нет необходимости записывать и решать дифференциальные уравнения, программа это сделает сама по предложенной ей исследователем структуре системы и параметрам ее элементов. Результаты решения выводятся в наглядной графической форме. Поэтому программой могут пользоваться и те, кто не имеет глубоких познаний в математике и программировании.

При использовании VisSim'a не требуется владеть программированием на языках высокого уровня или ассемблере. В то же время, специалисты, владеющие программированием, могут создавать собственные блоки, дополняя ими богатую библиотеку стандартных блоков VisSim'a.

1.1. Графический интерфейс VisSim

Интерфейс программы – это совокупность средств, позволяющих человеку общаться с ней:

- вводить и получать данные,
- контролировать ход выполнения компьютером программы,
- подавать управляющие воздействия и наблюдать реакцию на них программы и т. п.

Программа VisSim предоставляет исследователю графический интерфейс, позволяющий основную часть работы по созданию модели выполнить с помощью мыши, а параметры элементов ввести с клавиатуры. Интерфейс VisSim состоит из главного окна, имеющего меню и ряд кнопок управления и так называемого рабочего пространства, в котором строится и корректируется модель, наблюдаются результаты ее работы.

С точки зрения исследователя интерфейс программы VisSim представляет собой интерактивный виртуальный лабораторный стенд, обеспечивающий построение моделей из отдельных блоков, запуск процесса моделирования, управление им и контроль результатов [2].

Меню VisSim позволяет выполнять, наряду с другими, следующие важные действия:

- File (работа с файлами) – открывать новую диаграмму, сохранять, а затем при необходимости вновь открывать созданную диаграмму.
- Edit (Редактирование) – разворачивать на 180 градусов отдельные, выделенные блоки, а также создавать составные блоки и отменять последние изменения, внесенные в диаграмму.
- Simulate (Моделировать) – запускать процесс моделирования и изменять важные параметры этого процесса: время работы модели и величину шага интегрирования.
- Blocks (Блоки) – выносить на рабочее пространство элементарные блоки, из которых составляется модель, и надписи.
- Analyze (Анализировать) – строить частотные характеристики выделенных фрагментов линейной части модели.
- View (Вид) – устанавливать шрифт надписей и красивый вид блоков.
- Help (Справка) – получать справочный материал по программе и отдельным ее блокам.

Диаграммой в VisSim'e называется совокупность связанных, а также автономных блоков и надписей, помещенных на рабочее пространство и способных в известном смысле функционировать при запуске процесса моделирования. Диаграмма может быть сохранена в виде отдельного файла и, при необходимости, открыта вновь.

Модель VisSim’а – это часть диаграммы, содержащая виртуальный аналог реальной или проектируемой системы. Диаграмма может содержать несколько моделей.

1.2. Принципы построения моделей в среде VisSim

Исходными данными для построения модели в VisSim’е являются структурно-функциональная схема моделируемой системы, процесса или объекта и описывающие их дифференциально-алгебраические уравнения. Вместо таких уравнений могут быть заданы операторы или функции, характеризующие отдельные элементы моделируемой системы, например, передаточные функции для линейных элементов и статические характеристики для нелинейных элементов.

Реальные системы и объекты состоят из отдельных, связанных и взаимодействующих друг с другом элементов. И для всей системы в целом, и для ее отдельных элементов можно указать место приложения воздействия, которое можно назвать входом, и место их реакции на входное воздействие, называемое выходом. И воздействие, и реакция – это некоторые физические величины, являющиеся функциями времени.

Модели систем и объектов в программе VisSim строятся из отдельных элементов – блоков. Блок – это виртуальный аналог физического элемента реальной системы.

Взаимодействие между блоками отображается так называемыми линиями связи, указывающими направление передачи воздействий (сигналов) от одного блока к другому.

Взаимодействие между блоками моделируется сигналами – функциями времени, передаваемыми между блоками по линиям связи. Сигналы в модели могут быть измерены с помощью виртуальных измерительных устройств или рассмотрены и изучены с помощью виртуального осциллографа.

Принцип построения модели в VisSim'е состоит в вынесении на рабочее пространство моделей реальных элементов (блоков) и соединении их в соответствии с заранее составленной структурно-алгоритмической схемой моделируемой системы. Такое построение модели из виртуальных блоков очень похоже, с определенной степенью условности, на построение реальной системы из настоящих блоков в производственных условиях или на лабораторном стенде [1, 2].

1.3. Основные блоки VisSim

Элементарные блоки VisSim'а можно условно разделить на три основных категории и одну дополнительную:

- Блоки, имеющие только выход: генераторы.
- Блоки, имеющие вход и выход: преобразователи.
- Блоки, имеющие только вход: индикаторы.
- Блоки без входов и выходов: надписи, комментарии и др.

Для удобства работы с диаграммой часть ее блоков, в частности, часть блоков модели, можно заключить в один составной. Это позволяет создавать модели иерархической структуры очень сложных систем и объектов. Такие блоки могут иметь множество входов и выходов или не иметь их вообще.

Важным компонентом модели является соединительная линия – виртуальный аналог физического соединения элементов, передающего воздействия от одного элемента к другому. Соединительные линии в VisSim'е однонаправленные, передают сигналы с выхода одного блока к входу другого. Поэтому при построении модели следует так разделять реальную систему на функциональные блоки, чтобы последующий блок практически не влиял на функционирование предыдущего. Например, выходное электрическое сопротивление предыдущего блока должно быть значительно меньше входного сопротивления последующего блока.

Входные и выходные сигналы могут быть как одиночными функциями времени, так и набором таких функций. В таком случае сигнал называется векторным, как и соответствующий вход или выход блока.

Генераторы

Генераторы – это блоки, имеющие только выход. Генераторы вырабатывают изменяющиеся во времени или постоянные сигналы.

Примерами таких блоков в VisSim являются блоки:

- step (ступенька) – генератор ступенчатой единичной функции;
- ramp (спуск, подъем) – генератор линейно растущего сигнала;

- sinusoid – генератор синусоидального сигнала;
- const – генератор постоянного сигнала, величина которого не меняется в процессе работы модели;
- slider – генератор постоянного сигнала, величину которого можно менять в процессе работы модели.

Преобразователи

Преобразователи – это блоки, имеющие входы и выходы. Блоки-преобразователи воспринимают воздействия от других блоков, преобразовывают их в соответствии с определенными уравнениями или правилами и выдают преобразованный сигнал на выход.

Важнейшие блоки для моделирования линейных систем:

- transferFunction – передаточная функция. Этот блок позволяет создавать модели как простых, так и очень сложных линейных систем;
- integrator – блок интегратора, осуществляющий интегрирование входного сигнала по времени;
- summingJunction – сумматор двух и более сигналов, его выходной сигнал равен алгебраической сумме входных;
- gain – усилитель.

Индикаторы

Индикаторы – это блоки, имеющие только вход. Индикаторы программы VisSim предназначены для отображения сигналов в удобной и привычной для исследователя форме.

Важнейшими индикаторами являются блоки:

- осциллограф – plot.
  Виртуальный осциллограф VisSim’a представляет собой окно, похожее на экран осциллографа, в котором изображается зависимость наблюдаемых сигналов от времени. На боковой стороне осциллографа помещены условные изображения входов, к которым могут быть подключены выходы других блоков диаграммы для наблюдения поведения их сигналов в зависимости от времени.
- цифровой индикатор – display.
  Цифровой индикатор VisSim’a показывает в цифровом виде значение сигнала на выходе того блока, к которому он подключен. Этот прибор используется для измерения постоянных величин.

Надписи и комментарии

Надписи – это блоки без входов и выходов. Эти блоки позволяют создавать на рабочем пространстве диаграммы VisSim текстовые области, которые помогают понять смысл диаграммы и содержат сведения о том, кто, когда и какую диаграмму создал. Основной блок: label – надпись.

1.4. Понятие о принципах функционирования программы VisSim

После того, как модель построена, когда на рабочее пространство вынесены и соединены в нужном порядке блоки, составляющие систему, генераторы сигналов и индикаторы, а также введены параметры элементов модели, может быть запущен процесс ее функционирования. Для этого следует щелкнуть по кнопке «Пуск». Параметры некоторых сигналов и блоков исследователь может изменять в процессе работы модели, другие параметры можно изменить, остановив процесс работы модели. Продолжительность работы модели можно задавать до ее запуска, можно и прерывать работу модели по желанию исследователя.

Получив команду «Пуск», программа начинает анализировать то, как соединены блоки, на основе этого анализа составляет дифференциально-алгебраические уравнения, описывающие модель, и решает их. Полученные результаты, как функции модельного времени, периодически и очень часто, придаются значениям входных и выходных сигналов блоков.

Дифференциально-алгебраические уравнения математически описывают так называемые динамические объекты, объекты очень широкого класса, обладающие инерционностью и рядом других свойств. И поскольку программа VisSim способна решать такие уравнения, то в ней можно моделировать системы и объекты очень широкого диапазона сложности.

Решение уравнений проводится по шагам – дается малое приращение времени, вычисляются, с учетом начальных условий, значения сигналов на выходах и входах всех блоков, затем вновь дается малое приращение времени, проводятся вычисления и т. д. Малая величина шага интегрирования позволяет исследователю воспринимать сигналы как непрерывные. Выходные сигналы любого блока при желании можно наблюдать на экране виртуального осциллографа или измерять виртуальным цифровым индикатором. В результате решения можно получить зависимости выходных сигналов от времени. Таким образом, работа по моделированию систем в программе VisSim для исследователя похожа на работу на реальном стенде.

Кроме того, программа позволяет более глубоко проанализировать полученные результаты и оптимизировать модель. Например, VisSim предоставляет возможность быстрого построения частотных характеристик фрагментов модели и всей системы.

2. Анализ электронных и электрических схем (Pspice)

Программа **Pspice** рассчитывает следующие характеристики электронных цепей [3, 4]:

- режим цепи по постоянному току в «рабочей точке» (Bias Point);
- режим по постоянному току при вариации источников постоянного напряжения или тока, температуры и других параметров цепи (DC Sweep);
- чувствительность характеристик цепи к вариации параметров компонентов в режиме по постоянному току (Sensitivity);
- малосигнальные передаточные функции в режиме по постоянному току (Transfer Function);
- характеристики линеаризованной цепи в частотной области при воздействии одного или нескольких сигналов (AC Sweep);
- спектральную плотность внутреннего шума (Noise Analysis);
- переходные процессы при воздействии сигналов различной формы (Transient Analysis);
- спектральный анализ (Fourier Analysis);
- статистические испытания по методу Монте-Карло и расчет наихудшего случая (Monte Carlo/Worst Case);
- многовариантный анализ при вариации температуры (Temperature) и других параметров (Parametric).

1. AC Sweep – расчет частотных характеристик и уровня шума

.AC [LIN] [OCT] [DEC] *<n> <начальная частота> <конечная частота>*

Эта директива задает диапазон частот в пределах ***<начальная частота> ... <конечная частота>***.

Параметр ***LIN*** устанавливает линейный шаг по частоте, при этом ***n*** – общее количество точек по частоте.

Параметры ***OCT*** и ***DEC*** устанавливают логарифмический характер изменения частоты октавами и декадами соответственно. Параметр ***n*** определяет в таком случае количество точек по частоте на одной октаве или декаде.

.NOISE M*(<узел>[,<узел>]) <имя> <n>*

По этой директиве производится анализ спектральной плотности внутреннего шума.

Директива **.NOISE** указывается совместно с директивой **.AC**, в которой задается диапазон частот анализа. Источниками шума служат резисторы, ключи и полупроводниковые приборы. На каждой частоте рассчитывается спектральная плотность выходного напряжения, обусловленная наличием статистически независимых источников внутреннего шума. Точки съема выходного напряжения указываются по спецификации ***M(<узел> [,<узел>])***. К входным зажимам цепи подключается независимый источник напряжения или тока, ***<имя>*** которого приводится в списке параметров директивы **.NOISE**. Этот источник не является источником реального сигнала, он служит лишь для

обозначения входных зажимов цепи. Если ко входу подключается источник напряжения, то на входе рассчитывается эквивалентная спектральная плотность напряжения; если ко входу подключен источник тока, то рассчитывается эквивалентная спектральная плотность тока. Внутреннее сопротивление реального генератора сигнала должно быть включено в описание цепи как отдельный резистор. Если указан целочисленный параметр ***<n>***, то на каждой **n**-й частоте в диапазоне анализа будет рассчитываться не только спектральная плотность суммарного шума, но и вклад в нее каждого шумового источника.

Существует возможность задания следующих параметров режима **AC Sweep**:

1. характер изменения частоты:
   – *Linear* – линейная шкала;
   – *Octave* – изменение частоты октавами;
   – *Decade* – изменение частоты декадами;
2. параметры диапазона частот:
   – *Total Pts, Pts/Decade, Pts/Octave* – общее количество точек при выборе линейного масштаба или количество точек по частоте на одну декаду или октаву;
   – *Start Freq* – начальная частота;
   – *End Freq* – конечная частота.

Для расчета спектральной плотности внутреннего шума **Noise Analysis**:

– *Noise Enabled* – включение режима расчета уровня шума;
– *Output Voltage* – выходное напряжение;
– *I/ V Source* – имя входного источника напряжения или тока;
– *Interval* – интервал *n* расчета парциальных уровней шума.

Расчет характеристик в частотной области производится после определения режима по постоянному току и линеаризации нелинейных компонентов (это делается автоматически, никаких дополнительных директив не требуется). Все независимые источники напряжения V и тока I, для которых заданы параметры переменных сигналов (амплитуды и фазы), являются входными воздействиями.

Результаты расчета комплексных амплитуд узловых напряжений и токов ветвей выводятся по директивам **.PRINT**, **.PLOT** или **.PROBE**.

2. Load Bias Point – загрузка данных режима по постоянному току

.LOADBIAS <«*имя файла*»>

По этой директиве из файла, созданного в предыдущем сеансе работы с программой, считываются узловые потенциалы по постоянному току. Для выполнения расчета переходных процессов по директиве **.TRAN** с заданными начальными условиями этот файл можно предварительно отредактировать и заменить директиву **.NODESET** на **.IC**. Для передачи содержания файла узловых потенциалов, указанного в директиве **.LOADBIAS**, в выходной файл результатов (с расширением имени *.OUT), необходимо в директиве **.OPTIONS** указать параметр **EXPAND**.

3. Save Bias Point – сохранение данных режима по постоянному току

.SAVEBIAS <«имя файла»> <[OP] [DC] [TRAN]> [NOSUBCKT]
+ [TIME=<значение> [REPEAT]] [TEMP=<значение>] [STEP=<значение>]
+ [MCRUN=<значение>] [DC=<значение>] [DC1=<значение>]
+ [DC2=<значение>]

По этой директиве в файл с указанным именем записываются значения узловых потенциалов схемы для указанного вида анализа **OP**, **DC** или **TRAN**. Для каждого вида анализа в одном задании должна быть отдельная директива.

Параметр **NOSUBCKT** запрещает запись в файл узловых потенциалов внутренних узлов макромоделей.

Параметры **TIME=<значение> [REPEAT]** определяют моменты времени, в которые запоминаются узловые потенциалы схемы при расчете переходных процессов (режим **TRAN**). Ели параметр **REPEAT** не указан, то узловые потенциалы запоминаются в первый момент времени, равный указанному значению. При наличии параметра **REPEAT** параметр ***<значение>*** равен интервалу времени, с которым узловые потенциалы периодически запоминаются, при этом в файле сохраняется только последняя запись.

Параметр **TEMP=<значение>** задает температуру, для которой сохраняются узловые потенциалы при вариации температуры, а необязательный параметр **STEP=<значение>** задает интервал температур, через который обновляется запись в файл.

Параметр **MCRUN=<значение>** определяет количество вариантов расчетов, через которые обновляется запись в файл при статистическом анализе или расчете по методу наихудшего случая.

Параметры **DC = *<значение>*, DC1 = *<значение>*** и **DC2 = *<значение>*** задают значения параметров, варьируемых по директиве **.DC**, для которых производится запись в файл. Параметр **DC = *<значение>*** используется только при вариации одной переменной, а **DC1** и **DC2** – при вариации двух переменных (соответственно первой и второй).

Образуемый на диске текстовый файл имеет следующий формат. Сначала на одной или более строках в виде комментариев указываются имя схемы, заголовок, дата создания, затраты времени на расчет, тип анализа, температура и т. п. Далее помещается директива **.NODESET**, содержащая полную информацию об узловых потенциалах схемы. Поэтому после загрузки файла по директиве **.LOADBIAS** эти потенциалы будут установлены автоматически.

Одно из типичных применений директив **.SAVEBIAS**, **.LOADBIAS** – исследование больших схем, для которых расчет режима по постоянному току занимает значительное время. Тогда после расчета режима по постоянному току он может быть сохранен по директиве **.SAVEBIAS**, и в дальнейшем перед выполнением других видов анализа узловые потенциалы в этом режиме могут быть загружены по директиве **.LOADBIAS**.

4. DC Sweep – вариация параметров при расчете режима по постоянному току

Расчет режима по постоянному току производится при вариации одного или нескольких источников постоянного напряжения или тока, температуры, параметров моделей компонентов схемы и глобальных параметров по директивам

.DC [LIN] ***<имя 1-й переменной> <начальное значение>***
+ <конечное значение> <приращение>
+ [<имя 2-й переменной> <начальное значение>
+ <конечное значение> <приращение>]

.DC [OCT] [DEC] ***<имя 1-й переменной> <начальное значение> + <конечное значение> <количество точек>***
+ [<имя 2-й переменной> <начальное значение> + <конечное значение> <количество точек>]

.DC ***<имя 1-й переменной>*** **LIST** ***<значение> + [<имя 2-й переменной>*** **LIST** ***<значение>]***

Режим по постоянному току рассчитывается для нескольких значений переменных, в качестве которых могут приниматься:

- имена независимых источников напряжения или тока;
- параметры моделей компонентов (указываются тип компонента, имя модели и в круглых скобках имя варьируемого параметра);
- температура (в качестве ее имени указывается TEMP);
- глобальные параметры (указывается ключевое слово PARAM и вслед за ним имя варьируемого глобального параметра, определенного ранее).

Характер изменения переменных задается ключевыми словами:

- **LIN** – линейный масштаб (ключевое слово LIN можно не указывать);
- **DEC**, **OCT** – логарифмический масштаб декадами или октавами;
- **LIST** – список значений.

Если указаны спецификации двух варьируемых параметров, то первый параметр изменяется в заданных пределах для каждого значения второго параметра.

Максимальное количество итераций при переходе к следующему варианту по умолчанию устанавливается равным достаточно малой величине **ITL2=20**. Поэтому при необходимости можно увеличить значение **ITL2**, используя директиву **.OPTIONS**.

Тип варьируемого параметра:

- ***Voltage Source*** – источник напряжения;
- ***Temperature*** – температура;
- ***Current Source*** – источник тока;
- ***Model Parameter*** – параметр модели компонента;
- ***Global Parameter*** – глобальный параметр.

Тип вариации параметра:

- ***Linear*** – линейный масштаб;
- ***Octave*** – логарифмический масштаб октавами;
- ***Decade*** – логарифмический масштаб декадами;
- ***Value List*** – в виде списка параметров.

Пределы изменения параметров:

- ***Start Value*** – начальное значение;
- ***End Value*** – конечное значение;
- ***Increment*** – приращение;
- ***Pts/ Decade (Octave)*** – количество точек на одну декаду (октаву);
- ***Values*** – список параметров.

5. Monte Carlo/Worst Case – статистический анализ и наихудший случай

Статистический анализ по методу Монте-Карло (Monte Carlo) производится при статистическом разбросе параметров, описанных по директиве **.MODEL**.

Случайные величины создаются с помощью генераторов случайных чисел. Величина относительного разброса каждого параметра и закон распределения случайной величины задаются опцией директивы **.MODEL**.

Имеются генераторы случайных величин с двумя *стандартными законами распределения:*

- **UNIFORM** – равновероятное распределение на отрезке (-1, +1);
- **GAUSS** – гауссовское распределение на отрезке (-1, +1) с нулевым средним значением и среднеквадратическим отклонением 0,25.

Кроме того, пользователь может задать нестандартный закон распределения случайных величин с помощью директивы **.DISTRIBUTION.**

Случайным параметрам, закон распределения которых не задан явно в директиве **.MODEL**, по умолчанию назначается распределение, указанное в опции **DISTRIBUTION** директивы **.OPTIONS**.

Статистические испытания по методу Монте-Карло проводятся при расчете режима по постоянному току, переходных процессов или частотных характеристик по директиве

.MC <n> [DC] [IRAN] [AC] <*имя выходной переменой*>
+ <*обработка результатов*> [LIST]
+ [OUTPUT <*спецификация*>] [*NAME(<минимум>,<максимум>)*]
+ [SEED=<значение>]

Параметр ***<n>*** задает количество статистических испытаний. Ключевые слова **DC**, **TRAN**, **AC** указывают вид анализа. После них указывается ***<имя выходной переменной>****,* подлежащей статистической обработке.

При статистическом анализе предусматривается разнообразная статистическая обработка результатов моделирования, характер которой определяется с помощью опции ***<обработка результатов>****,* принимающей одно из следующих значений:

- **YMAX** – расчет максимального отклонения текущей реализации от номинальной;
- **MAX** – расчет максимального значения в каждой реализации;
- **MIN** – расчет минимального значения в каждой реализации;
- ***RlSE_EDGE(<значение>)*** – определение момента первого пересечения заданного уровня снизу вверх;

- ***FALL_EDGE(<значение>)*** – определение момента первого пересечения заданного уровня сверху вниз.

По необязательному ключевому слову **LIST** на печать выводится список значений всех случайных параметров во всех реализациях.

В отсутствие ключевого слова **OUTPUT** характеристики цепи, указанные в директивах **.PRINT**, **.PLOT** или **.PROBE**, выводятся на печать или передаются в постпроцессор Probe один раз для номинального значения случайных параметров. С помощью ключевого слова **OUTPUT** их можно вывести требуемое число раз, задавая после этого слова следующие параметры:

- **ALL** – во всех реализациях;
- **FIRST <*m*>** – для первых *m* реализаций;
- **EVERY <*m*>** – на каждой m-й реализации;
- **RUNS <*m1*>,<*m2*>...** – для реализаций с указанными номерами *m1, m2,...*

После ключевого слова **RANGE** определяется диапазон значений, в пределах которого статистически обрабатывается выходная переменная.

Переменная **SEED** задает начальное значение датчика случайных чисел. Оно может принимать нечетные значения в диапазоне 1...32767. По умолчанию SEED = 17533. При одинаковых значениях **SEED** результаты статистического моделирования при последующих запусках программы повторяются.

Расчет чувствительности и наихудшего случая (WorstCase)

Для этого применяется директива

WCASE [DC][TRAN][AC] <*имя выходной переменной*>
+ <*обработка результатов*> [<*опции*>]

Проводятся расчеты характеристик цепи при вариации параметров. Сначала по очереди изменяются все указанные параметры, что позволяет оценить параметрическую чувствительность характеристик. В заключение рассчитываются характеристики цепи при одновременном изменении всех параметров по методу наихудшего случая.

Ключевые слова **HI** и **LOW** задают направление изменения параметров компонентов относительно номинальных значений при расчете наихудшего случая. Если определяется функция **YMAX** или **MAX**, по умолчанию назначается ключевое слово **HI** (положительное приращение), в противном случае – **LOW** (отрицательное приращение).

Ключевые слова **VARY DEV**, **VARY LOT**, **VARY BOTH** определяют характер случайного разброса параметров.

Ключевые слова **BY RELTOL**, **BYE <*значение*>** задают относительное изменение параметров при расчете чувствительности. По умолчанию их изменение равно значению параметра **RELTOL,** указанному в директиве **.OPTIONS**.

Ключевое слово **DEVICES(< *список типов компонентов*>)** назначает типы компонентов, параметры которых изменяются при расчете чувствительности и наихудшего случая. Список типов компонентов состоит из первых символов их имен, перечисляемых слитно, без пробелов. Например,

вариация параметров только резисторов (R), конденсаторов (C) и биполярных транзисторов (Q) назначается с помощью ключевого слова **DEVICES RCQ**.

6. Bias Point Detail – вывод подробной информации о режиме по постоянному току

Режим по постоянному току всегда рассчитывается вначале моделирования перед выполнением других видов анализа без указания специальных директив. При расчете режима по постоянному току принимаются во внимание параметры всех независимых источников напряжения и тока. Результаты расчетов выводятся в текстовый файл *.OUT в виде таблицы узловых потенциалов и списка токов независимых источников. При наличии директивы **.OP** дополнительно рассчитываются малосигнальные параметры линеаризованных схем замещения полупроводниковых приборов и нелинейных управляемых источников, которые также выводятся в выходной файл *.OUT.

Кроме того, анализ по постоянному току выполняется перед расчетом переходных процессов по директиве **.TRAN** для определения начальных условий (если отсутствует ключевое слово **SKIPBP**) и перед анализом в частотной области по директиве **.AC** для линеаризации нелинейных компонентов в окрестности режима по постоянному току.

Приближенные значения режима по постоянному току с помощью директивы **.NODESET** нужно указывать при анализе схем, имеющих несколько устойчивых состояний.

7. Digital Setup – задание параметров цифровых устройств

Тип времени запаздывания во всех компонентах:

- ***Minimum*** – минимальное;
- ***Typical*** – типичное;
- ***Maximum*** – максимальное;
- ***Worst-case (Min/Max)*** – вариация задержки при расчете наихудшего случая (минимальная/максимальная).

Начальные значения выходным состояниям триггеров:

- ***All X*** – присвоить неопределенное состояние X;
- ***All 0*** – присвоить состояние логического «О»;
- ***All 1*** – присвоить состояние логической «1».

8. Options – параметры моделирования

Параметры и режимы работы программы Pspice устанавливаются в диалоговом окне, открываемом нажатием кнопки **Options** в окне выбора директив моделирования. Это выполняется с помощью директивы

.OPTIONS *[имя опции]* [<имя опции> = <значение>]*

Опции перечисляются в любом порядке.

Они подразделяются на два вида:

- опции, имеющие численное значение;
- опции, не имеющие численного значения; их можно назвать флагами, находящимися в положении «включено» (Y) или «выключено» (N).

Список флагов (в скобках указаны значения по умолчанию):

- **ACCT** – вывод статистики времени выполнения всех видов анализа, характеристик цепи и других данных о задании на моделирование (N);
- **EXPAND** – включение в описание схемы описания макромоделей (N);
- **LIBRARY** – включение в описание схемы описания моделей из библиотечных файлов (N);
- **LIST** – вывод списка всех компонентов цепи (N);
- **NOBIAS** – запрещение вывода в выходной файл значений узловых потенциалов в рабочей точке (N);
- **NODE** – печать списка соединений (N);
- **NOECHO** – запрещение включения в выходной файл части описания схемы, располагаемой после строки с директивой **.OPTIONS** (N);
- **NOICTRANSLATE** – отмена установки начальных условий расчета переходных процессов, выполненных с помощью директив **.IC** (имеются в виду начальные напряжения на конденсаторах и токи через индуктивности) (N);
- **NOMOD** – запрещение вывода списка параметров моделей (N);
- **NOOUTMSG** – подавление передачи в выходной файл сообщений об ошибках моделирования (N);
- **NOPAGE** – запрещение перевода страниц в выходном файле (N);
- **NOPRBMSG** – подавление передачи в файл данных для программы Probe сообщений об ошибках моделирования (N);
- **NOREUSE** – запрещение автоматического сохранения и восстановления информации о режиме по постоянному току при вариации температуры, статистическом анализе, расчете наихудшего случая и при вариации параметров (N);
- **OPTS** – вывод значений всех опций (N);
- **STEPGMIN** – включение алгоритма расчета режима по постоянному току вариацией проводимости GMIN в случае отсутствия сходимости метода Ньютона-Рафсона. При наличии этой опции в отсутствии сходимости сначала применяется метод вариации GMIN и затем, в случае неудачи, метод вариации источников питания (в отсутствии этой опции используется только метод вариации источников питания) (N).

Таблица 1. Опции, имеющие численные значения

Имя опции	Наименование	Размер-ность	Значение по умолчанию
ABSTOL	Допустимая ошибка расчета токов в режиме TRAN	А	10^{-12}
CHGTOL	Допустимая ошибка расчета заряда в режиме TRAN	Кл	10^{-14}
CPTIME	Максимальное время работы процессора, разрешенное для выполнения данного задания	с	0
DEFAD	Диффузионная площадь стока МОП-транзистора (AD)	$м^2$	0
DEFAS	Диффузионная площадь истока МОП-транзистора (AS)	$м^2$	0
DEFL	Длина канала МОП-транзистора (L)	м	10^{-4}
DEFW	Ширина канала МОП-транзистора (W)	м	10^{-4}
DIGDRVF	Минимальное выходное сопротивление цифровых устройств (для моделей UIO)	Ом	2
DIGDRVZ	Максимальное выходное сопротивление цифровых устройств (для моделей UIO)	кОм	20
DIGERRDEFAULT	Максимальное количество контролируемых ошибок цифровых устройств		20
DIGERRLIMIT	Максимальное количество сообщений об ошибках в цифровых устройствах		0**
DIGFREQ	Частота дискретизации при анализе цифровых устройств	Гц	10^{10}
DIGINITSTATE	Установка начального состояния триггеров: 0 – сброс; 1 – установка; 2 – X		2
DIGIOLVL	Уровень интерфейса А/Ц, Ц/А по умолчанию		1
DIGMNTYMX	Селектор выбора задержки цифрового устройства по умолчанию: 1 – минимум; 2 – типичное значение; 3 – максимум; 4 – мин/макс		2
DIGMNTYSCALE	Масштабный коэффициент для расчета минимальной задержки		0,4
DIGTYMXSCALE	Масштабный коэффициент для расчета максимальной задержки		1,6

Продолжение таблицы 1

Имя опции	Наименование	Размер-ность	Значение по умолчанию
DIGOVRDRV	Отношение выходных сопротивлений цифровых устройств, при которых изменяется состояние общего выходного узла		3
DISTRIBUTION	Закон распределения отклонений параметров от номинальных значений		UNIFORM
GMIN	Минимальная проводимость ветви цепи (проводимость ветви, меньшая GMIN, считается равной нулю)	См	10^{-12}
ITL1	Максимальное количество итераций в режиме DC		150
ITL2	Максимальное количество итераций при расчете передаточных функций по постоянному току при переходе к последующей точке		20
ITL4	Максимальное количество итераций при переходе к следующему моменту времени в режиме TRAN		10
ITL5	Общее максимальное количество всех итераций в режиме TRAN (установка ITL5 = 0 означает бесконечность)		0
LIMPTS	Максимальное количество точек, выводимых в таблицу или на график		0
NUMDGT	Количество значащих цифр в таблицах выходных данных (не более 8)		4
PIVREL	Относительная величина элемента строки матрицы, необходимая для его выделения в качестве ведущего элемента (режим AC)		10^{-3}
PIVTOL	Абсолютная величина элемента строки матрицы, необходимая для его выделения в качестве ведущего элемента (режим AC)		10^{-13}
RELTOL	Допустимая относительная ошибка расчета напряжений и токов в режиме TRAN		10^{-3}
TNOM	Номинальная температура	°C	27
VNTOL	Допустимая ошибка расчета напряжений в режиме TRAN	В	10^{-6}
WIDTH	Длина строки выходного файла (аналогично директиве .WIDTH)		80

В процессе моделирования программа Pspice генерирует различные сообщения, которые передаются в выходной файл и файл данных для программы Probe.

Статистические сведения о задании выводятся в выходной файл с расширением .OUT при введении опции **ACCT** в директиве **.OPTION**.

Таблица 2. Данные, помещаемые в выходном файле

Параметр	**Значение**
NUNODS	Количество узлов схемы устройства без учета подсхем
NCNODS	Количество узлов схемы устройства с учетом подсхем
NUMNOD	Общее количество узлов схемы замещения устройства с учетом внутренних узлов встроенных моделей компонентов
NUMEL	Общее количество компонентов устройства с учетом подсхем
DIODES	Количество диодов с учетом подсхем
BJTS	Количество биполярных транзисторов с учетом подсхем
JFETS	Количество полевых транзисторов с учетом подсхем
MFETS	Количество МОП-транзисторов с учетом подсхем
GASFETS	Количество арсенид-галлиевых полевых транзисторов с учетом подсхем
IGBTS	Количество статически индуцированных биполярных транзисторов с учетом подсхем
NDIGITAL	Количество цифровых устройств с учетом подсхем
NSTOP	Размерность воображаемой матрицы цепи (фактически не все элементы разреженных матриц хранятся в памяти)
NTTAR	Фактическое количество входов в матрице цепи в начале вычислений
NTTBR	Фактическое количество входов в матрице цепи в конце вычислений
NTTOV	Количество ненулевых элементов матрицы цепи
IFILL	Разность между NTTAR и NTTBR
IOPS	Количество операций с плавающей запятой, необходимых для решения одного матричного уравнения цепи
PERSPA	Степень разреженности матрицы цепи в процентах
NUMTTP	Количество шагов интегрирования переходного процесса
NUMRTP	Количество моментов времени при расчете переходного процесса, при которых шаг интегрирования был слишком велик и расчет повторен с меньшим шагом
NUMNIT	Общее количество итераций при расчете переходного процесса
DIGTP	Количество временных шагов при логическом моделировании

Продолжение таблицы 2

Параметр	**Значение**
DIGEVT	Количество логических событий
DIGEVL	Количество вычислений логических состояний цифровых устройств
MEMUSE	Размер используемой области ОЗУ в байтах
Matrix solution	Время, затраченное на решение матричного уравнения
Matrix load	Время, затраченное на составление уравнений компонентов
READIN	Время, затраченное на чтение входного файла и поиск ошибок в нем
SETUP	Время, затраченное на формирование матрицы цепи
DC sweep	Время, затраченное на расчет передаточных функций по постоянному току
Bias point	Время, затраченное на расчет режима по постоянному току в рабочей точке
Digital simulation	Время, затраченное на вычисление логических состояний цифровых устройств
AC and noise	Время, затраченное на расчет в частотной области
Transient analysis	Время, затраченное на расчет переходного процесса
Monte Carlo	Время, затраченное на выполнение директив .MC и .WCASE
OUTPUT	Время, затраченное на переформатирование данных, необходимое перед выполнением директив .PRINT и .PLOT
OVERHEAD	Время, затраченное на выполнение задания
Total job time	Общее время выполнения задания, за исключением времени, затраченного на загрузку файлов программы Pspice

9. Parametric – многовариантный анализ

Вариация параметров назначается по директиве **.STEP**, имеющей следующие разновидности:

.STEP [LIN] ***<имя варьируемого параметра> <начальное значение>*** **+** ***<конечное значение> <шаг приращения параметра>***

.STEP [OCT] [DEC] ***<имя варьируемого параметра>***

+ ***<начальное значение> <конечное значение> <количество точек>***

.STEP ***<имя варьируемого параметра>*** **LIST <** ***значение>***

На каждом шаге вариации параметров по очереди выполняются все виды анализа характеристик цепи, задаваемых директивами **.DC**, **.AC**, **.TRAN** и др. Варьироваться могут все параметры всех моделей компонентов и глобальные параметры за исключением:

- параметров L и W МОП-транзистора (разрешается варьировать аналогичные параметры LD и WD);
- температурных коэффициентов TCI, TC2 резисторов и других компонентов.

10. Sensitivity – чувствительность в режиме по постоянному току

Чувствительность в режиме малого сигнала рассчитывается по директиве

.SENS <*выходная переменная*>

Чувствительность рассчитывается после линеаризации цепи в окрестности рабочей точки. По директиве **.SENS** рассчитывается чувствительность каждой из указанных выходных переменных к изменению параметров всех компонентов и моделей. Поэтому объем результатов расчета чувствительностей может быть очень большим.

Результаты расчета выводятся в файл .out. Выходные переменные указываются по тому же формату, что и в директивах **.PRINT** для режимов **TRAN** и **DC**. При этом накладывается ограничение: если выходная переменная должна быть током, то допускается только ток через независимые источники напряжения.

11. Temperature – вариация температуры

Вариация температуры производится по директиве

.TEMP <*температура*>

Здесь указывается список значений температуры (по шкале Цельсия), для которых следует выполнить все указанные в задании директивы анализа характеристик. Если указано несколько значений температуры, то все виды анализа проводятся для каждой температуры. Если директива **.TEMP** не приведена, а в директиве **.OPTIONS** не указано другого значения температуры, то расчеты проводятся для номинальной температуры Tnom = 27 °C.

12. Transfer Function – передаточные функции по постоянному току

Малосигнальные передаточные функции в режиме по постоянному току рассчитываются по директиве

.TF <*выходная переменная*> <*имя источника напряжения или тока*>

Они рассчитываются после линеаризации цепи в окрестности рабочей точки. Выходные переменные имеют тот же формат, что и по директиве **.PRINT**. Если выходная переменная должна быть током, то это ток через независимый источник напряжения. В качестве входной переменной может быть использовано напряжение или ток источника напряжения или тока. Результаты расчетов выводятся в выходной файл *.OUT без обращения к директивам **.PRINT** или **.PLOT**.

13. Transient – расчет переходных процессов

Переходные процессы рассчитываются по директиве

.TRAN[/OP] <*шаг вывода данных*> <*конечное время*>

+ [<*начальный момент времени вывода данных*>

+ [<*максимальный шаг*>]] [SKIPBP]

Переходные процессы всегда рассчитываются с момента $t = 0$ до момента ***<конечное время>.*** Перед началом расчета переходных процессов рассчитывается режим по постоянному току. Шаг интегрирования выбирается автоматически. Максимальное значение шага интегрирования устанавливается параметром ***<максимальный шаг>***; если он не указан, то максимальный шаг

интегрирования устанавливается равным ***<конечное время>*/50**. Если исследуемая схема не имеет инерционностей, то шаг интегрирования равен величине ***<шаг вывода данных>***.

Величина ***<шаг вывода данных>*** используется для вывода данных по директивам **.PRINT** и **.PLOT**. При этом для расчета значений переменных применяется квадратичная интерполяция между дискретными. С ***<шагом вывода данных>*** рассчитываются с помощью обратного преобразования Лапласа импульсные характеристики управляемых источников, заданных передаточными функциями.

Если задан параметр ***<начальный момент времени вывода данных>,*** то вывод результатов расчета подавляется на интервале времени от t = 0 до указанного значения.

Режим по постоянному току определяет начальные условия для расчета переходных процессов.

Если в конце директивы **.TRAN** указать параметр **SKIPBP** (Skip Bias Point), то расчет режима по постоянному току отменяется. При этом начальные значения напряжений на емкостях и токов через индуктивности указываются в опциях вида IC= ..., включенных в описания конденсаторов и индуктивностей, а начальные значения узловых потенциалов указываются в директиве **.IC**.

Спектральный анализ проводится по директиве

.FOUR <*частота первой гармоники f1*> [*количество гармоник*]

+ <*выходная переменная*>

Спектральный анализ производится с помощью быстрого преобразования Фурье после завершения расчета переходного процесса (в задании на моделирование должна иметься и директива **.TRAN**). Имена переменных, спектр которых должен быть рассчитан, указываются в списке ***<выходная переменная>***. В директиве **.FOUR** задается частота первой гармоники и количество гармоник. Максимальное количество гармоник n = 100. По умолчанию рассчитываются первые 9 гармоник. В программе рассчитываются амплитуды постоянной составляющей и остальных n гармоник.

Спектральному анализу подвергается участок реализации переходного процесса длительностью T = 1/f1 в конце интервала анализа (чтобы завершились переходные процессы).

Результаты спектрального анализа выводятся в выходной файл *.OUT в виде таблиц без указания директив **.PRINT**, **.PLOT** или **.PROBE**. Кроме того, рассчитывается коэффициент нелинейных искажений (в процентах).

Для повышения точности расчета спектров рекомендуется с помощью параметра ***<максимальный шаг>*** задать максимальное значение шага интегрирования, равное требуемой величине шага дискретизации по времени.

14. Задание начальных условий

Начальные значения узловых потенциалов по постоянному току задаются по директиве

.IC ***V(<номер узла>[,<номер узла>]) = Значение ЭДС***

К указанным узлам подключаются источники постоянного напряжения с внутренним сопротивлением 0,0002 Ом, и рассчитывается режим по постоянному току. После завершения расчета эти источники отключаются – так задаются начальные значения узловых потенциалов перед расчетом переходных процессов.

Если в задании имеются и директива **.NODESET**, и директива **.IC**, то первая не будет выполняться при расчете режима по постоянному току перед началом анализа переходных процессов.

Задание начального приближения узловых потенциалов по постоянному току производится по директиве

.NODESET ***<V(<узел>[,<узел>])=<значение ЭДС>>***

Эта директива назначает начальное значение указанных потенциалов на нулевой итерации при расчете режима по постоянному току как в режиме **DC**, так и при расчете переходных процессов (в режиме **DC Sweep** она выполняется только на первом шаге варьирования источников напряжения). Если заданные значения узловых потенциалов близки к точному решению, то процесс итерационного расчета режима по постоянному току завершается за меньшее количество итераций. Эта директива полезна при расчете очень больших схем по частям и расчете схем с несколькими устойчивыми состояниями.

15. Управление выдачей результатов

Результаты расчетов в виде таблиц выводятся в выходной файл с расширением *.OUT по директиве

.PRINT ***[/DGTLCHG] [DC] [IRAN] [AC] [NOISE] <выходная переменная>***

В одной директиве **.PRINT** можно выбрать только один вид анализа и привести список не более восьми выходных переменных. Одновременно в задании на моделирование можно поместить несколько таких директив. В таблицах каждая колонка соответствует одной переменной. В первой колонке помещается независимая переменная: постоянное напряжение (режим DC), время (режим TRAN) или частота (режим AC).

Количество значащих цифр данных и максимальное количество строк в таблице определяются опциями **NUMDGT** и **LIMPTS** директивы **.OPTIONS**.

Логические состояния цифровых компонентов выводятся на внешние устройства после окончания моделирования обычным образом. В общем случае спецификация ***<выходная переменная>*** цифровых узлов имеет вид:

D(<имя узла>)

При наличии параметра **/DGTLCHG** префикс **D** можно опустить. Различие этих директив заключается в том, что по директиве **.PRINT** на печать выводятся состояния как цифровых, так и аналоговых узлов, а по директиве **.PRINT/DGTLCHG** – только цифровых.

Результаты в виде графиков выводятся в выходной файл по директиве
.PLOT ***[DC] [AC] [NOISE] [TRAN] <выходная переменная>***
+ [(<нижняя граница>,<верхняя граница>)]

Смысл параметров такой же, что и в директиве **.PRINT**. Графики выводятся с помощью буквенно-цифровых символов независимо от типа печатающего устройства. Однако директивой **.PLOT** пользоваться на современных ПК не имеет смысла. На одном графике помещается до восьми кривых, причем количество директив **.PLOT** в одном задании не ограничено. Диапазон по оси X указан в директиве, устанавливающей вид анализа, а диапазон по оси Y определяется с помощью параметров ***<нижняя граница>***, ***<верхняя граница>*** или автоматически.

Графический постпроцессор Probe подключается директивой
.PROBE ***[/CSDF] [<выходная переменная>]***

Если список выходных переменных не указан, то в файл результатов с расширением имени .DAT заносятся потенциалы всех узлов цепи и токи всех компонентов, разрешенных для помещения в список выходных переменных.

В файл данных *.DAT всегда помещаются уровни внутреннего шума **INOISE**, **ONOISE** и данные о кривых гистерезиса магнитных сердечников B(H), поэтому при наличии списка выходных переменных их в него включать не надо.

По директиве **.PROBE/CSDF** создается файл результатов в текстовом виде с расширением имени *.TXT, который можно использовать для сопряжения с программами дополнительной обработки результатов.

3. Моделирование систем электропривода (Matlab Simulink)

Одним из индустриальных стандартов является пакет **Matlab** с широко развитыми дополнениями, из которых **Simulink** наиболее приспособлен для анализа и синтеза различных систем [5].

Пакет Simulink со своими дополнениями – основной инструмент изучения различных электромеханических систем. Практически нет ни одной задачи, связанной с исследованием систем электропривода, которую нельзя было бы решить в этом пакете.

Simulink предоставляет исследователю самые различные возможности, начиная от структурного (математического) представления системы и кончая генерированием кодов для программирования микропроцессора в соответствии со структурной схемой модели [6].

Simulink – это интерактивная система для моделирования динамических систем. Она представляет собой среду, которая позволяет моделировать процесс путем перетаскивания блоков диаграмм на экране и их манипуляцией. Simulink работает с линейными, нелинейными, непрерывными, дискретными, многомерными системами [7].

Основные средства для моделирования и анализа, имеющиеся в пакете Simulink:

- обширная библиотека блоков для создания линейных и нелинейных, дискретных и непрерывных, гибридных моделей;
- иерархическая структура моделей с неограниченной вложенностью;
- скалярные и векторные связи;
- средство для создания пользовательских блоков и библиотек;
- интерактивное моделирование с "живым" отображением на экране;
- семь методов интегрирования с фиксированным и переменным шагом;
- линеаризация;
- определение точек равновесия;
- различные способы вывода на экран и библиотека входных сигналов.

Программа **Simulink** является приложением к пакету **MATLAB**. При моделировании с использованием **Simulink** реализуется принцип визуального программирования, в соответствии с которым пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет расчеты.

Simulink является достаточно самостоятельным инструментом **MATLAB** и при работе с ним не требуется знать сам **MATLAB** и остальные его приложения. С другой стороны доступ к функциям **MATLAB** и другим его инструментам остается открытым и их можно использовать в **Simulink**.

При моделировании пользователь может выбирать метод решения дифференциальных уравнений, а также способ изменения модельного времени (с фиксированным или переменным шагом). В ходе моделирования имеется возможность следить за процессами, происходящими в системе. Для этого используются специальные устройства наблюдения, входящие в состав

библиотеки **Simulink**. Результаты моделирования могут быть представлены в виде графиков или таблиц.

Имеются обширные библиотеки блоков для разных областей применения. При работе с **Simulink** пользователь имеет возможность модернизировать библиотечные блоки, создавать свои собственные, а также составлять новые библиотеки блоков. **Simulink** позволяет пополнять библиотеки блоков с помощью подпрограмм, написанных на языке **MATLAB**, **C + +**, и **Fortran**.

3.1. Создание модели

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

1. Создать новый файл модели с помощью команды **File/New/Model**.
2. Расположить блоки в окне модели.
3. Если требуется, изменить параметры блоков.
4. После установки на схеме всех блоков из требуемых библиотек нужно выполнить соединение элементов схемы.

3.2. Установка параметров расчета и его выполнение

Перед выполнением расчетов необходимо предварительно задать параметры расчета. Задание параметров расчета выполняется в панели управления меню **Simulation/Parameters**.

Окно настройки параметров расчета имеет 4 вкладки:

1. **Solver** (Расчет) – Установка параметров расчета модели.
2. **Workspace I/O** (Ввод/вывод данных в рабочую область) – Установка параметров обмена данными с рабочей областью **MATLAB**.
3. **Diagnostics** (Диагностика) – Выбор параметров диагностического режима.
4. **Advanced** (Дополнительно) – Установка дополнительных параметров.

Solver

Установка параметров расчета модели выполняется с помощью элементов управления, размещенных на вкладке **Solver**. Эти элементы разделены на три группы:

1. Simulation time (Интервал моделирования):

Время расчета задается указанием начального (**Start time**) и конечного (**Stop time**) значений времени расчета.

2. Solver options (Параметры расчета):

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

Список методов расчета нового состояния системы содержит несколько вариантов. Вариант **discrete** используется для расчета дискретных систем. Остальные методы используются для расчета непрерывных систем.

При выборе **Fixed-step** необходимо указать величину шага моделирования. Величина шага моделирования по умолчанию устанавливается системой автоматически (**auto**). Требуемая величина шага может быть введена вместо значения **auto** либо в форме числа, либо в виде вычисляемого выражения.

При выборе **Fixed-step** необходимо также задать режим расчета (**Mode**). Для параметра **Mode** доступны три варианта:

- **MultiTasking** (Многозадачный) – необходимо использовать, если в модели присутствуют параллельно работающие подсистемы, и результат работы модели зависит от временных параметров этих подсистем. Режим позволяет выявить несоответствие скорости и дискретности сигналов, пересылаемых блоками друг другу.
- **SingleTasking** (Однозадачный) – используется для тех моделей, в которых недостаточно строгая синхронизация работы отдельных составляющих не влияет на конечный результат моделирования.
- **Auto** (Автоматический выбор режима) – позволяет **Simulink** автоматически устанавливать режим **MultiTasking** для тех моделей, в которых используются блоки с различными скоростями передачи сигналов, и режим **SingleTasking** для моделей, в которых содержатся блоки, оперирующие одинаковыми скоростями.

При выборе **Variable-step** необходимо задать следующие параметры:

- **Max step size** – максимальный шаг расчета. По умолчанию он устанавливается автоматически (**auto**), и его значение в этом случае равно **(StopTime – StartTime)/50**. Довольно часто это значение оказывается слишком большим, и наблюдаемые графики представляют собой ломаные (а не плавные) линии. В этом случае величину максимального шага расчета необходимо задавать явным образом.
- **Min step size** – минимальный шаг расчета.
- **Initial step size** – начальное значение шага моделирования.

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

3. Output options (Параметры вывода).

Для данного параметра возможен выбор одного из трех вариантов:

- **Refine output** (Скорректированный вывод) – позволяет изменять дискретность модельного времени и тех сигналов, которые сохраняются в рабочей области **MATLAB** с помощью блока **To Workspace**. Установка величины дискретности выполняется, используя параметр **Refine factor**. По умолчанию его значение равно **1**, это означает, что регистрация производится с шагом **dt=1** (то есть для каждого значения модельного времени). Если задать **Refine factor** равным 2, это означает, что будет регистрироваться каждое второе значение сигналов, 3 – каждое третье т. д.

- **Produce additional output** (Дополнительный вывод) – обеспечивает дополнительную регистрацию параметров модели в заданные моменты времени; их значения задаются в виде списка.
- **Produce specified output only** (Формировать только заданный вывод) – устанавливает вывод параметров модели только в заданные моменты времени, которые указываются в параметре **Output times** (Моменты времени вывода).

Workspace I/O

Элементы, позволяющие управлять вводом и выводом в рабочую область **MATLAB** промежуточных данных и результатов моделирования, расположены на вкладке **Workspace I/O**.

Элементы вкладки разделены на 3 поля:

1. Load from workspace (Загрузить из рабочей области).

Если флажок **Input** (Входные данные) установлен, то можно ввести формат данных, которые будут считываться из рабочей области **MATLAB**.

Установка флажка **Initial State** (Начальное состояние) позволяет ввести в связанном с ним текстовом поле имя переменной, содержащей параметры начального состояния модели.

Данные, указанные в полях **Input** и **Initial State**, передаются в исполняемую модель посредством одного или более блоков **In** (из раздела библиотеки **Sources**).

2. Save to workspace (Записать в рабочую область) – позволяет установить режим вывода значений сигналов в рабочую область **MATLAB** и задать их имена.

3. Save options (Параметры записи) – задает количество строк при передаче переменных в рабочую область.

Если флажок **Limit rows to last** установлен, то в поле ввода можно указать количество передаваемых строк (отсчет строк производится от момента завершения расчета). Если флажок не установлен, то передаются все данные.

Параметр **Decimation** (Исключение) задает шаг записи переменных в рабочую область.

Параметр **Format** (формат данных) задает формат передаваемых в рабочую область данных. Доступные форматы **Array** (Массив), **Structure** (Структура), **Structure With Time** (Структура с дополнительным полем – «время»).

Diagnostics

Вкладка **Diagnostics** позволяет изменять перечень диагностических сообщений, выводимых **Simulink** в командном окне **MATLAB**, а также устанавливать дополнительные параметры диагностики модели.

Сообщения об ошибках или проблемных ситуациях, обнаруженных **Simulink** в ходе моделирования и требующих вмешательства разработчика, выводятся в командном окне **MATLAB**. Исходный перечень таких ситуаций и вид реакции на них приведен в списке на вкладке **Diagnostics**. Разработчик

может указать вид реакции на каждое из них, используя группу переключателей в поле **Action**:

None – игнорировать,

Warning – выдать предупреждение и продолжить моделирование,

Error – выдать сообщение об ошибке и остановить сеанс моделирования.

Выполнение расчета

Запуск расчета выполняется с помощью выбора пункта меню **Simulation/Start**.

Процесс расчета можно завершить досрочно, выбрав пункт меню **Simulation/Stop** или инструмент.

Расчет также можно остановить (**Simulation/Pause**) и затем продолжить (**Simulation/Continue**).

3.3. Редактор дифференциальных уравнений DEE

Simulink содержит специальный блок – **Differential Equation Editor** (редактор дифференциальных уравнений). С помощью этого блока можно задавать системы дифференциальных уравнений в явной форме и выполнять их решение.

Вызов редактора выполняется вводом команды **dee** в окне **MATLAB**.

Использование редактора рассмотрим на примере расчета переходных процессов в последовательном колебательном контуре. Задача заключается в нахождении тока, протекающего в электрической цепи и напряжения на конденсаторе С после замыкания ключа. Схема цепи показана на рис. 3.1. Начальные условия полагаем нулевыми (ток в цепи отсутствует, и конденсатор не заряжен).

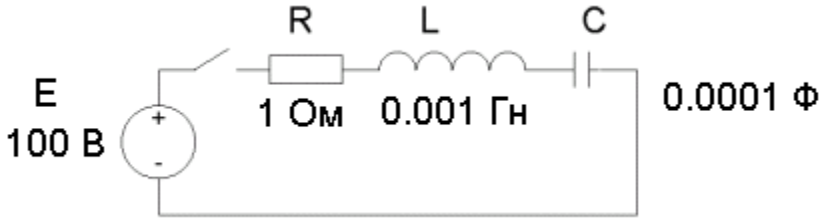


Рис. 3.1.

Предварительно составляем систему дифференциальных уравнений, описывающую электрическую цепь,

$$e = L\frac{di}{dt} + R\cdot i + u_C,\ i = c\frac{du_C}{dt},$$

где i – ток в цепи, u_C – напряжение на конденсаторе.

Записываем данную систему уравнений в явной форме

$$\frac{di}{dt} = (e - R\cdot i - u_C)/L,\ \frac{du_C}{dt} = \frac{1}{C}i.$$

Вводим «машинные» переменные

$$i \to x(1),\ u_C \to x(2),\ e \to u(1).$$

В итоге система уравнений примет вид

$$\frac{dx(1)}{dt} = (u(1) - R \cdot x(1) - x(2))/L\,,\ \frac{dx(2)}{dt} = \frac{1}{C}x(1)\,.$$

Введение «машинных» переменных связано с тем, что редактор дифференциальных уравнений требует задавать в виде векторов входные воздействия (**u**) и переменные состояния (**x**), и имена этих векторов жестко заданы.

После получения системы дифференциальных уравнений с использованием «машинных» переменных необходимо запустить редактор дифференциальных уравнений и ввести систему дифференциальных уравнений, начальные условия, а также алгебраические уравнения для расчета выходных сигналов (в рассматриваемой задаче выходные переменные равны переменным состояния – x(1), x(2)). Также необходимо указать размерность вектора входного сигнала (# of inputs – 1).

3.4. Sources – источники сигналов

1. Источник постоянного сигнала Constant.

Назначение:

Задает постоянный по уровню сигнал.

Параметры:

1. **Constant value** – Постоянная величина.
2. **Interpret vector parameters as 1-D** – Интерпретировать вектор параметров как одномерный (при установленном флажке).

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

2. Источник синусоидального сигнала Sine Wave.

Назначение:

Формирует синусоидальный сигнал с заданной частотой, амплитудой, фазой и смещением.

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

- **Time-based** – По текущему времени.
- **Sample-based** – По величине шага модельного времени.

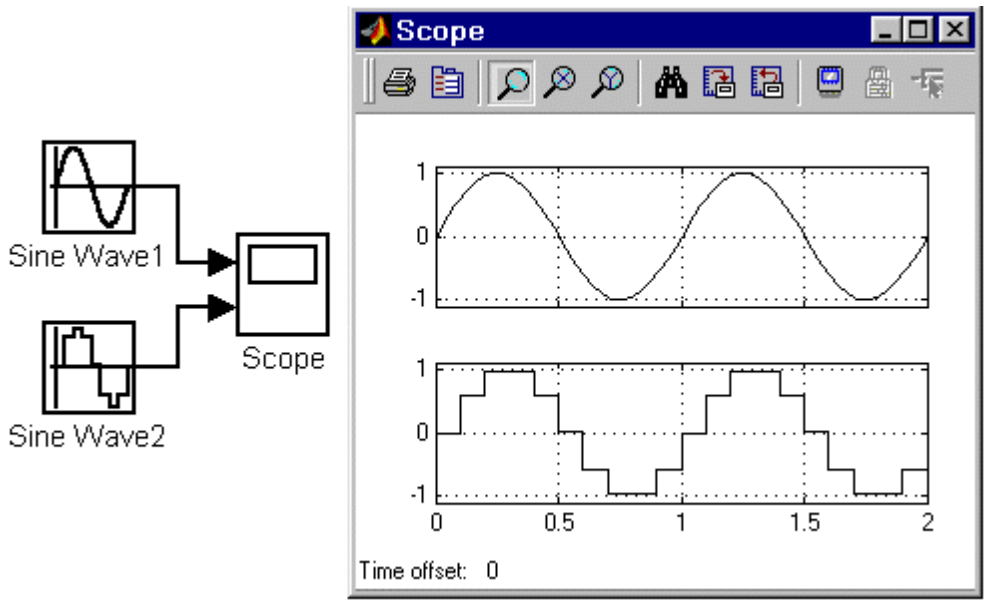


Рис. 3.2.

Формирование выходного сигнала по текущему значению времени для непрерывных систем.

Выходной сигнал источника в этом режиме соответствует выражению

y = Amplitude·sin(frequency·time + phase) + bias.

Параметры:

1. **Amplitude** – Амплитуда.
2. **Bias** – Постоянная составляющая сигнала.
3. **Frequency** (rads/sec) – Частота (рад/с).
4. **Phase** (rads) – Начальная фаза (рад).
5. **Sample time** – Шаг модельного времени. Параметр может принимать следующие значения:
   - **0** (по умолчанию) – Используется при моделировании непрерывных систем.
   - **>0** (положительное значение) – Задается при моделировании дискретных систем. В этом случае шаг модельного времени можно интерпретировать как шаг квантования по времени выходного сигнала.
   - **-1** – Шаг модельного времени устанавливается таким же, как и в предшествующем блоке, т. е. блоке, откуда приходит сигнал в данный блок.

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

Формирование выходного сигнала по величине модельного времени и количеству расчетных шагов на один период.

Выходной сигнал источника в этом режиме соответствует выражению

y= Amplitude·sin[(k+Number of offset samples)/Samples per period] + bias,

где **k** – номер текущего шага расчета.

Параметры:

1. **Amplitude** – Амплитуда.
2. **Bias** – Постоянная составляющая сигнала.
3. **Samples per period** – Количество расчетных шагов на один период синусоидального сигнала.
4. **Samples per period = 2p / (frequency·Sample time)**.
5. **Number of offset samples** – Начальная фаза сигнала. Задается количеством шагов модельного времени.
6. **Number of offset samples = Phase·Samples per period/ (2p)**.
7. **Sample time** – Шаг модельного времени.

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

3. Источник линейно изменяющегося воздействия Ramp.

Назначение:

Формирует линейный сигнал вида **y = Slope·time + Initial value**.

Параметры:

1. **Slope** – Скорость изменения выходного сигнала.
2. **Start time** – Время начала формирования сигнала.
3. **Initial value** – Начальный уровень сигнала на выходе блока.

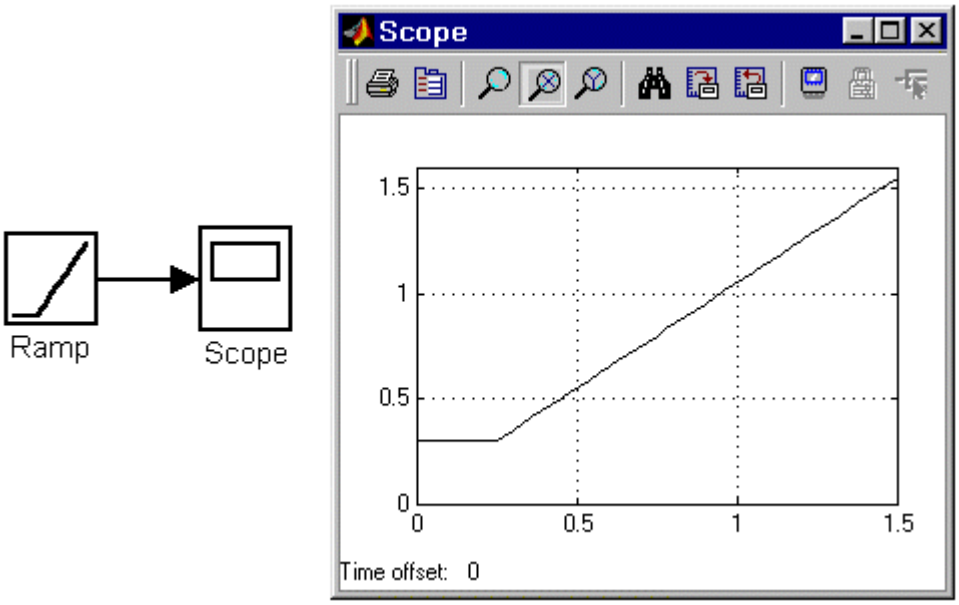


Рис. 3.3.

4. Генератор ступенчатого сигнала Step.

Назначение:

Формирует ступенчатый сигнал.

Параметры:

1. **Step time** – Время наступления перепада сигнала (с).
2. **Initial value** – Начальное значение сигнала.
3. **Final value** – Конечное значение сигнала.

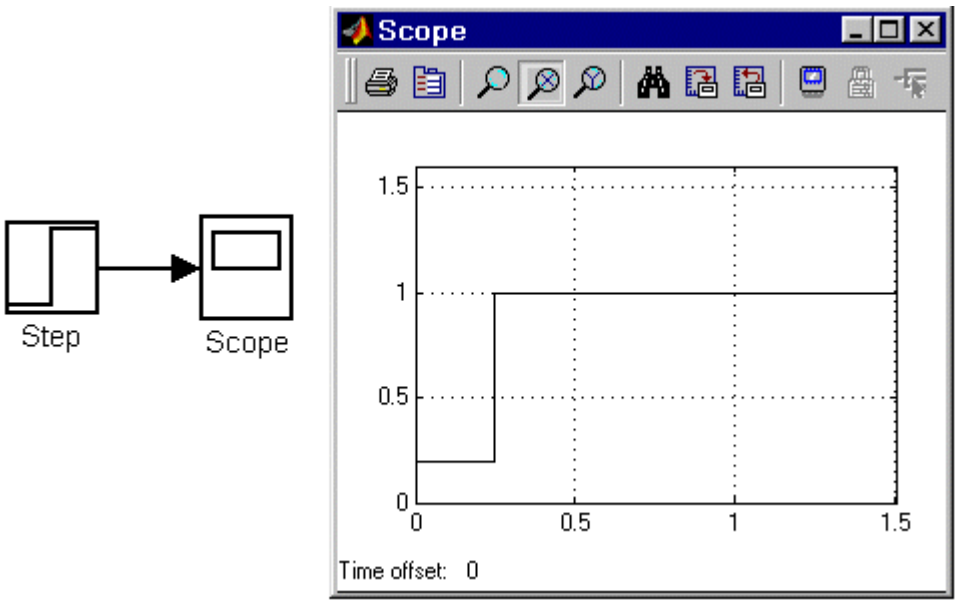


Рис. 3.4.

5. Генератор сигналов Signal Generator .

Назначение:

Формирует один из четырех видов периодических сигналов:

- **sine** – Синусоидальный сигнал.
- **square** – Прямоугольный сигнал.
- **sawtooth** – Пилообразный сигнал.
- **random** – Случайный сигнал.

Параметры:

1. **Wave form** – Вид сигнала.
2. **Amplitude** – Амплитуда сигнала.
3. **Frequency** – Частота (рад/с).
4. **Units** – Единицы измерения частоты. Может принимать два значения: Hertz – Гц и rad/sec – рад/с.

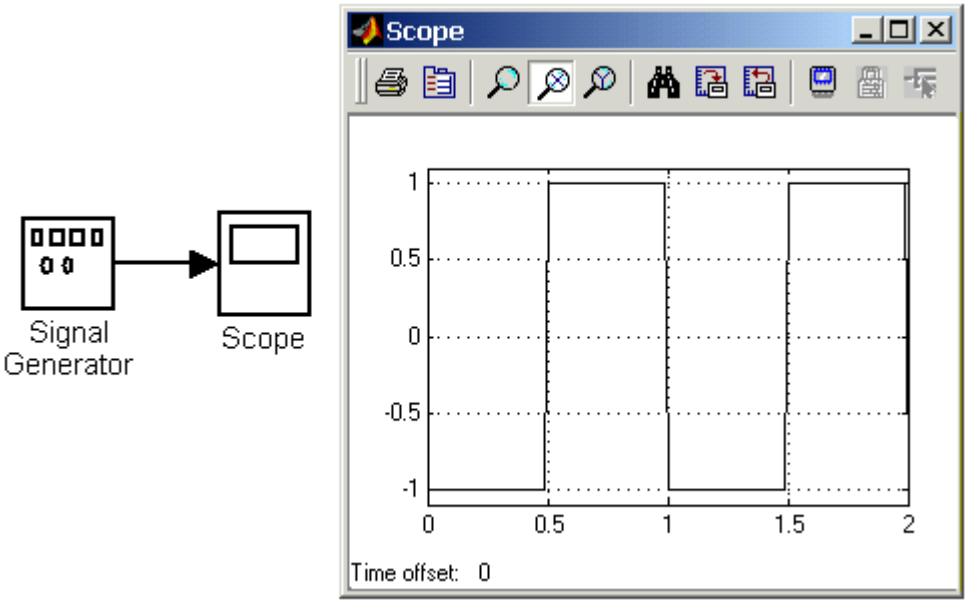


Рис. 3.5.

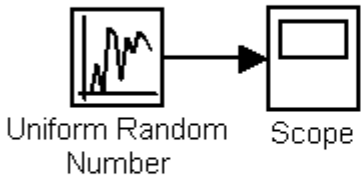
6. Источник случайного сигнала с равномерным распределением Uniform Random Number.

Назначение:

Формирование случайного сигнала с равномерным распределением.

Параметры:

1. **Minimum** – Минимальный уровень сигнала.
2. **Maximum** – Максимальный уровень сигнала.
3. **Initial seed** – Начальное значение.



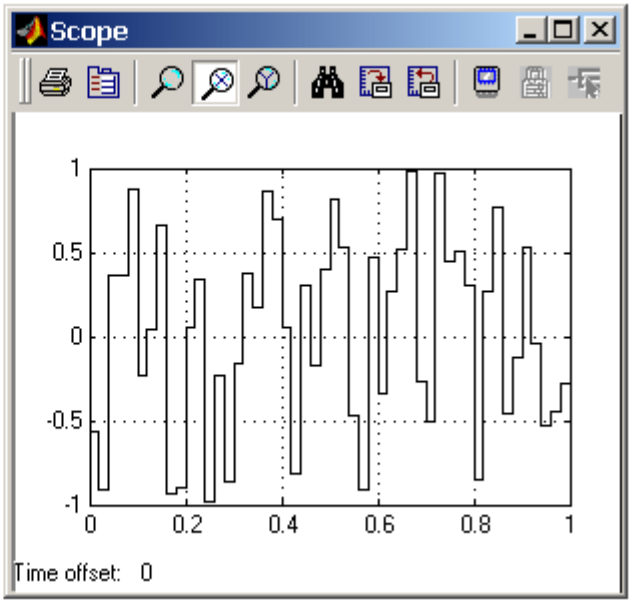


Рис. 3.6.

7. Источник случайного сигнала с нормальным распределением Random Number.

Назначение:

Формирование случайного сигнала с нормальным распределением уровня сигнала.

Параметры:

1. **Mean** – Среднее значение сигнала.
2. **Variance** – Дисперсия (среднеквадратическое отклонение).
3. **Initial seed** – Начальное значение.

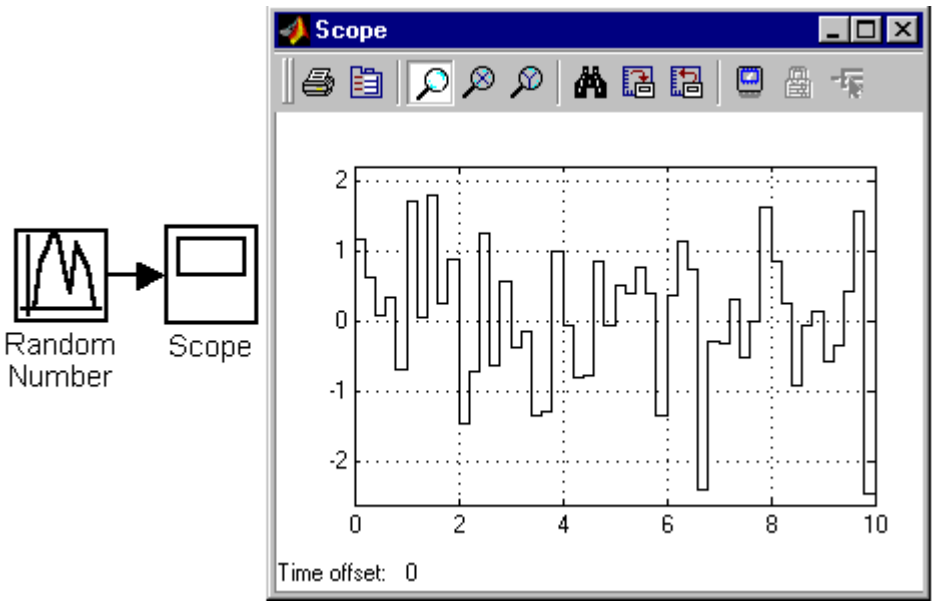


Рис. 3.7.

8. Источник импульсного сигнала Pulse Generator.

Назначение: Формирование прямоугольных импульсов.

Параметры:

1. **Pulse Type** – Способ формирования сигнала. Может принимать два значения: **Time-based** (по текущему времени) и **Sample-based** (по величине модельного времени и количеству расчетных шагов).
2. **Amplitude** – Амплитуда.
3. **Period** – Период.
4. **Pulse width** – Ширина импульсов.
5. **Phase delay** – Фазовая задержка.
6. **Sample time** – Шаг модельного времени.

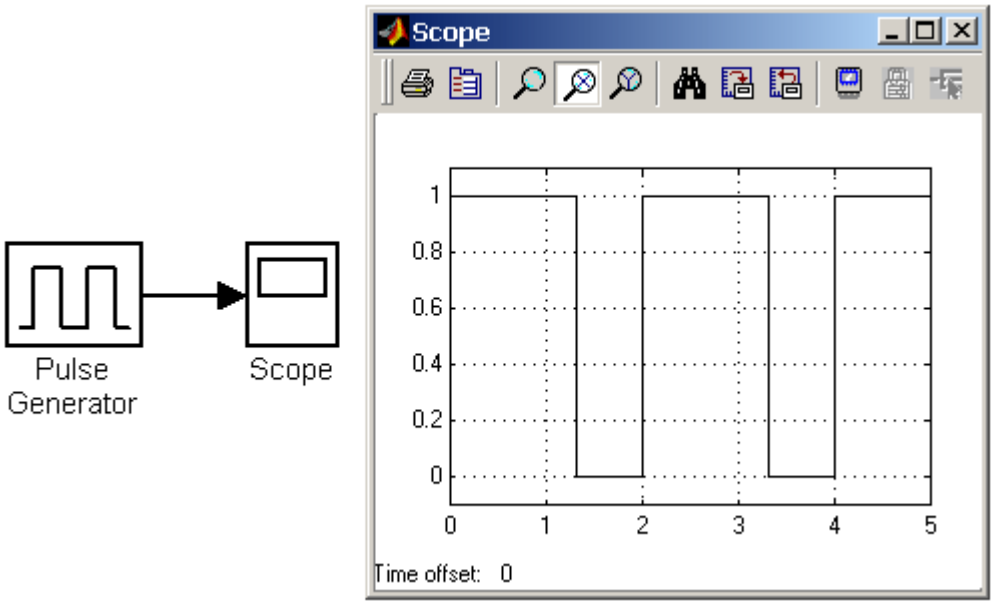


Рис. 3.8.

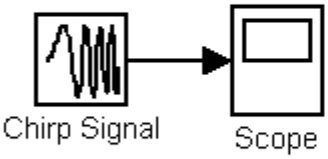
9. Генератор линейно-изменяющейся частоты Chirp Generator.

Назначение:

Формирование синусоидальных колебаний, частота которых линейно изменяется.

Параметры:

1. **Initial frequence** – Начальная частота (Гц).
2. **Target time** – Время изменения частоты (с).
3. **Frequence at target time** – Конечное значение частоты (Гц).



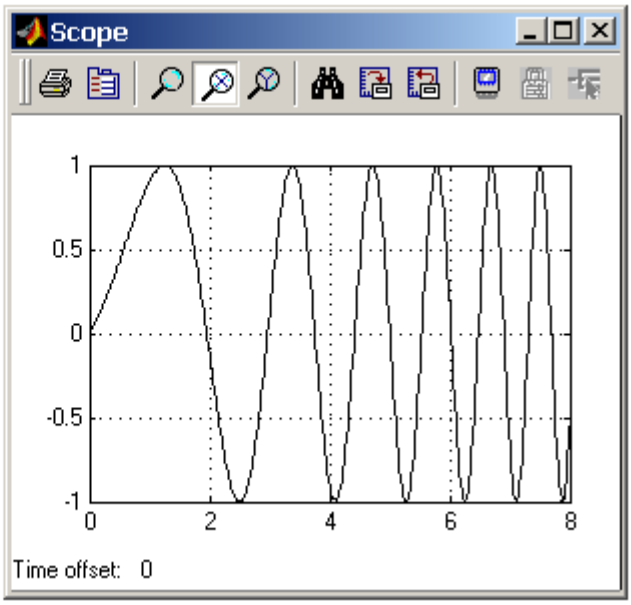


Рис. 3.9.

10. Генератор белого шума Band-Limited White Noice.

Назначение:

Создает сигнал заданной мощности, равномерно распределенной по частоте.

Параметры:

1. **Noice Power** – Мощность шума.
2. **Sample Time** – Модельное время.
3. **Seed** – Число, необходимое для инициализации генератора случайных чисел.

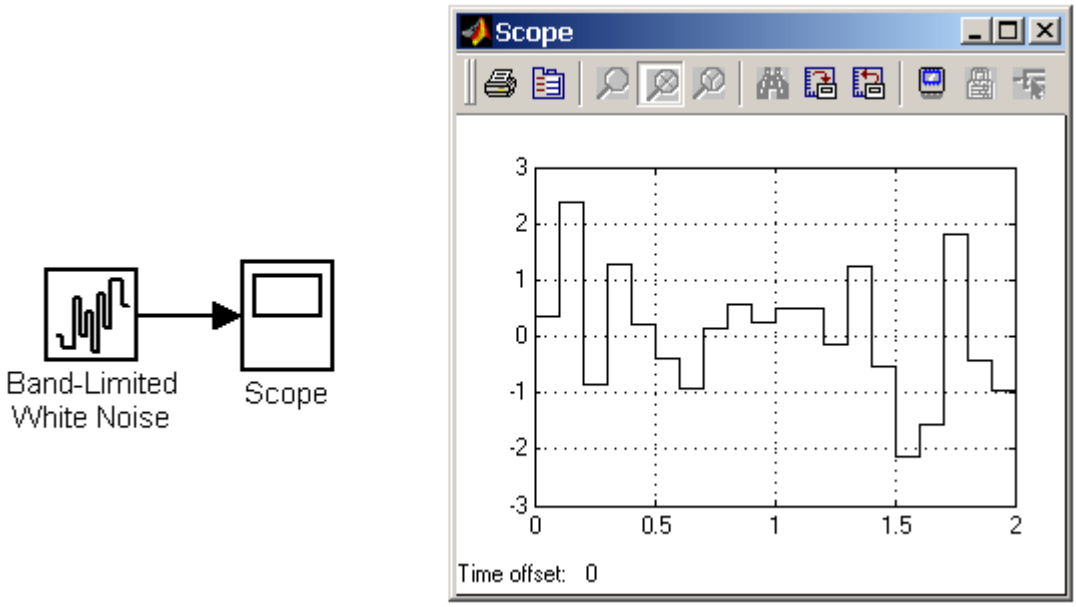


Рис. 3.10.

11. Источник временного сигнала Clock.

Назначение:

Формирует сигнал, величина которого на каждом шаге расчета равна текущему времени моделирования.

Параметры:

1. **Decimation** – Шаг, с которым обновляются показания времени на изображении источника (в том случае, если установлен флажок параметра **Display time**). Параметр задается как количество шагов расчета.
2. **Display time** – Отображение значения времени в блоке источника.

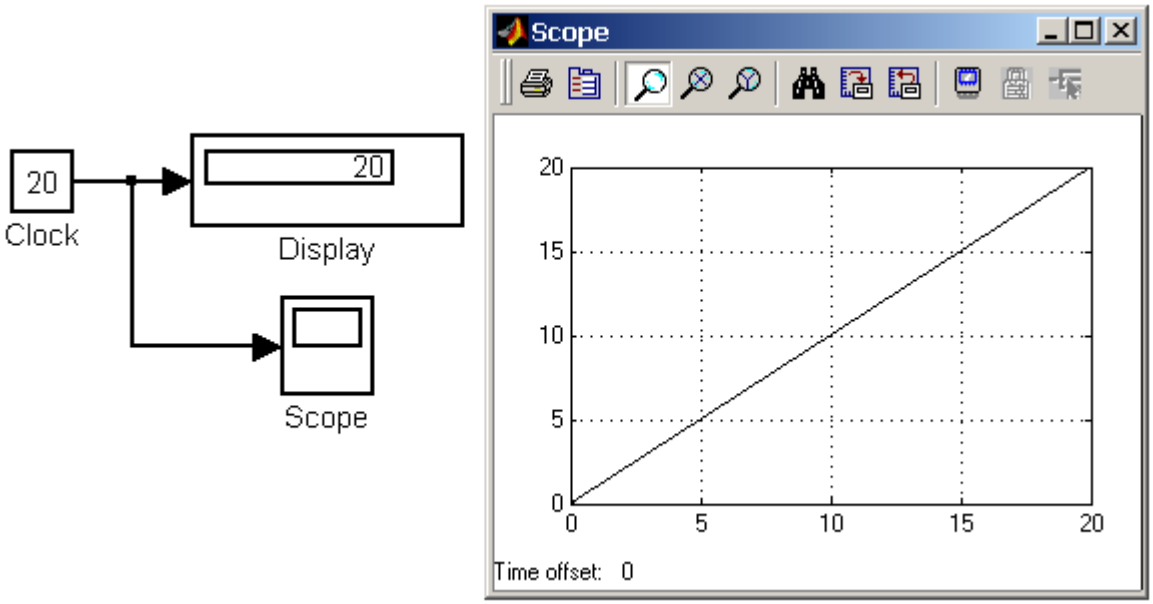


Рис. 3.11.

12. Цифровой источник времени Digital Clock.

Назначение:

Формирует дискретный временной сигнал.

Параметр:

1. **Sample time** – Шаг модельного времени (с).

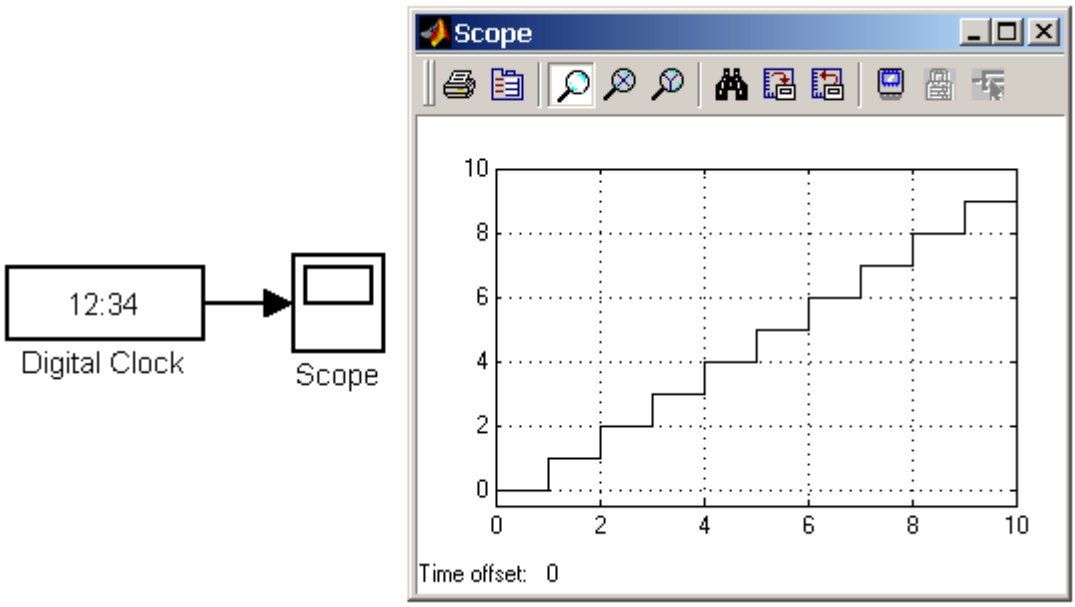


Рис. 3.12.

13. Блок считывания данных из файла From File.

Назначение:

Получение данных из внешнего файла.

Параметры:

1. **File Name** – Имя файла с данными.
2. **Sample time** – Шаг изменения выходного сигнала блока.

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

Файл данных (mat-файл), из которого считываются значения, не является текстовым. Структура файла подробно описана в справочной системе **MATLAB**. Пользователям **Simulink** удобнее всего создавать mat-файл с помощью блока **To File** (библиотека **Sinks**).

14. Блок считывания данных из рабочего пространства From Workspace.

Назначение:

Получение данных из рабочего пространства **MATLAB**.

Параметры:

1. **Data** – Имя переменной (матрицы или структуры), содержащей данные.
2. **Sample time** – Шаг изменения выходного сигнала блока.
3. **Interpolate data** – Интерполяция данных для значений модельного времени, не совпадающих со значениями в переменной Data.
4. **Form output after final data value by** – Вид выходного сигнала по окончании значений времени в переменной **Data**:
   - **Extrapolate** – Линейная экстраполяция сигналов.
   - **SettingToZero** – Нулевые значения сигналов.
   - **HoldingFinalValue** – Выходные значения сигналов равны последним значениям.
   - **CyclicRepetition** – Циклическое повторение значений сигналов. Данный вариант может использоваться, только если переменная **Data** имеет формат **Structure without time**.

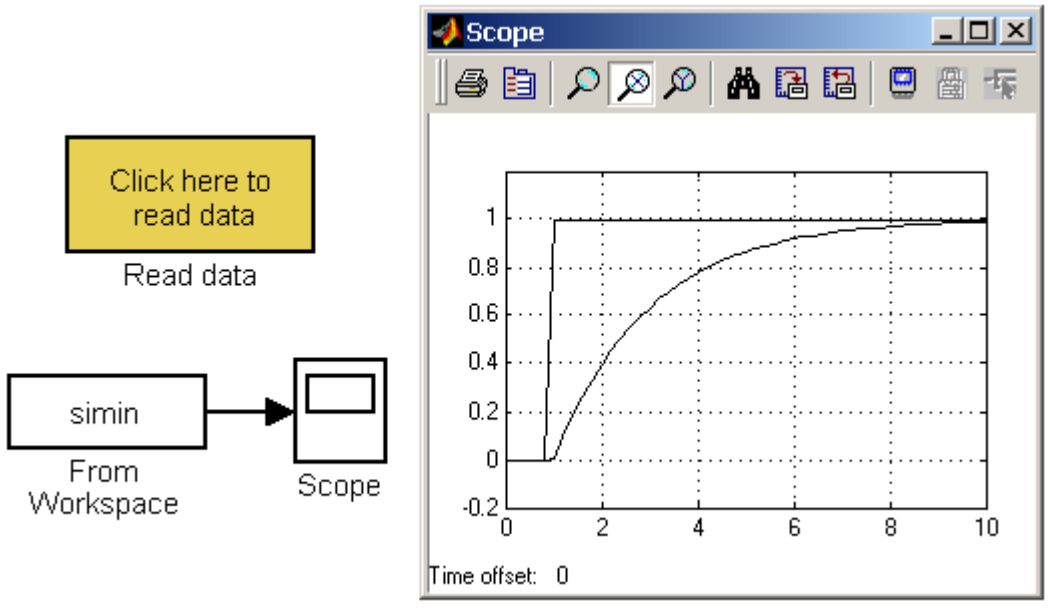


Рис. 3.13.

15. Блок сигнала нулевого уровня Ground.

Назначение:

Формирование сигнала нулевого уровня.

Параметры:

Нет.

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

16. Блок периодического сигнала Repeating Sequence.

Назначение:

Формирование периодического сигнала.

Параметры:

1. **Time values** – Вектор значений модельного времени.
2. **Output values** – Вектор значений сигнала для моментов времени, заданных вектором **Time values**.

Блок выполняет линейную интерполяцию выходного сигнала для моментов времени, не совпадающих со значениями заданными вектором **Time values**. На рис. 3.14 показан пример использования блока для формирования пилообразного сигнала. Значения модельного времени заданы вектором [0 3], а значения выходного сигнала – вектором [0 2].

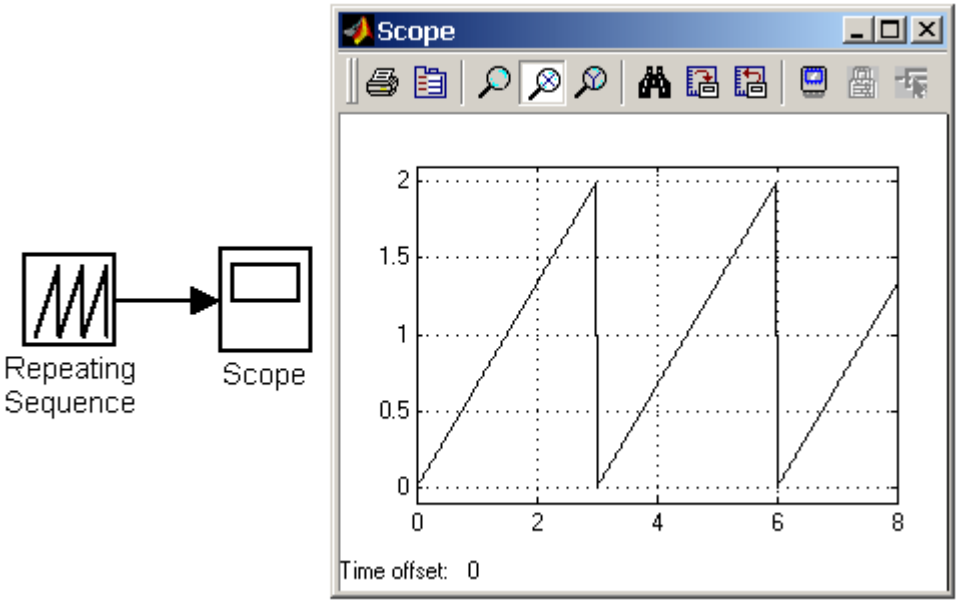


Рис. 3.14.

17. Блок входного порта Inport.

Назначение:

Создает входной порт для подсистемы или модели верхнего уровня.

Параметры:

1. **Port number** – Номер порта.
2. **Port dimensions** – Размерность входного сигнала. Если этот параметр равен **–1**, то размерность входного сигнала будет определяться автоматически.
3. **Sample time** –Шаг модельного времени.
4. **Data type** – Тип данных входного сигнала: auto, double, single, int8, uint8, int16, uint16, int32, uint32 или boolean.
5. **Signal type** – Тип входного сигнала:
   - **auto** – Автоматическое определение типа.
   - **real** –Действительный сигнал.
   - **Complex** – Комплексный сигнал.

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

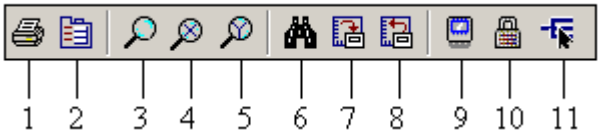
3.5. Sinks – приемники сигналов

1. Осциллограф Scope.

Назначение:

Строит графики исследуемых сигналов в функции времени. Позволяет наблюдать за изменениями сигналов в процессе моделирования.

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



Настройка окна осциллографа выполняется с помощью панелей инструментов. Панель инструментов содержит 11 кнопок:

1. **Print** – печать содержимого окна осциллографа.
2. **Parameters** – доступ к окну настройки параметров.
3. **Zoom** – увеличение масштаба по обеим осям.
4. **Zoom X-axis** – увеличение масштаба по горизонтальной оси.
5. **Zoom Y-axis** – увеличение масштаба по вертикальной оси.
6. **Autoscale** – автоматическая установка масштабов по обеим осям.
7. **Save current axes settings** – сохранение текущих настроек окна.
8. **Restore saved axes settings** – установка ранее сохраненных настроек окна.
9. **Floating scope** – перевод осциллографа в «свободный» режим.
10. **Lock/Unlock axes selection** – закрепить/разорвать связь между текущей координатной системой окна и отображаемым сигналом. Инструмент доступен, если включен режим **Floating scope**.
11. **Signal selection** – выбор сигналов для отображения. Инструмент доступен, если включен режим **Floating scope**.

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

- **General** – общие параметры.
- **Data history** – параметры сохранения сигналов в рабочей области **MATLAB**.

На вкладке **General** задаются следующие параметры:

1. **Number of axes** – число входов (систем координат) осциллографа. При изменении этого параметра на изображении блока появляются дополнительные входные порты.
2. **Time range** – величина временного интервала, для которого отображаются графики. Если время расчета модели превышает заданное параметром **Time range**, то вывод графика производится порциями, при этом интервал отображения каждой порции графика равен заданному значению **Time range**.
3. **Tick labels** – вывод/скрытие осей и меток осей. Может принимать три значения (выбираются из списка):
   - **all** – подписи для всех осей,
   - **none** – отсутствие всех осей и подписей к ним,
   - **bottom axis only** – подписи горизонтальной оси только для нижнего графика.
4. **Sampling** – установка параметров вывода графиков в окне. Задает режим вывода расчетных точек на экран. При выборе **Decimation** кратность вывода устанавливается числом, задающим шаг выводимых расчетных точек. Маркерами на графиках отмечены расчетные точки.
5. **Floating scope** – перевод осциллографа в «свободный» режим (при установленном флажке).

На вкладке **Data history** задаются следующие параметры:

1. **Limit data points to last** – максимальное количество отображаемых расчетных точек графика. При превышении этого числа начальная часть графика обрезается. В том случае, если флажок параметра **Limit data points to last** не установлен, то **Simulink** автоматически увеличит значение этого параметра для отображения всех расчетных точек.
2. **Save data to workspace** – сохранение значений сигналов в рабочей области **MATLAB**.
3. **Variable name** – имя переменной для сохранения сигналов в рабочей области **MATLAB**.
4. **Format** – формат данных при сохранении в рабочей области **MATLAB**. Может принимать значения:
   - **Array** – массив,
   - **Structure** – структура,
   - **Structure with time** – структура с дополнительным полем «время».

2. Осциллограф Floating Scope.

Осциллограф **Floating Scope**, по сути, есть обычный осциллограф **Scope**, переведенный в «свободный» режим.

В этом режиме блок осциллографа не имеет входов, а выбор отображаемого сигнала осуществляется с помощью инструмента (**Signal selection**) панели инструментов.

3. Графопостроитель XY Graph.

Назначение:

Строит график одного сигнала в функции другого (график вида Y(X)).

Параметры:

1. **x-min** – Минимальное значение сигнала по оси X.
2. **x-max** – Максимальное значение сигнала по оси X.
3. **y-min** – Минимальное значение сигнала по оси Y.
4. **y-max** – Максимальное значение сигнала по оси Y.
5. **Sample time** – шаг модельного времени.

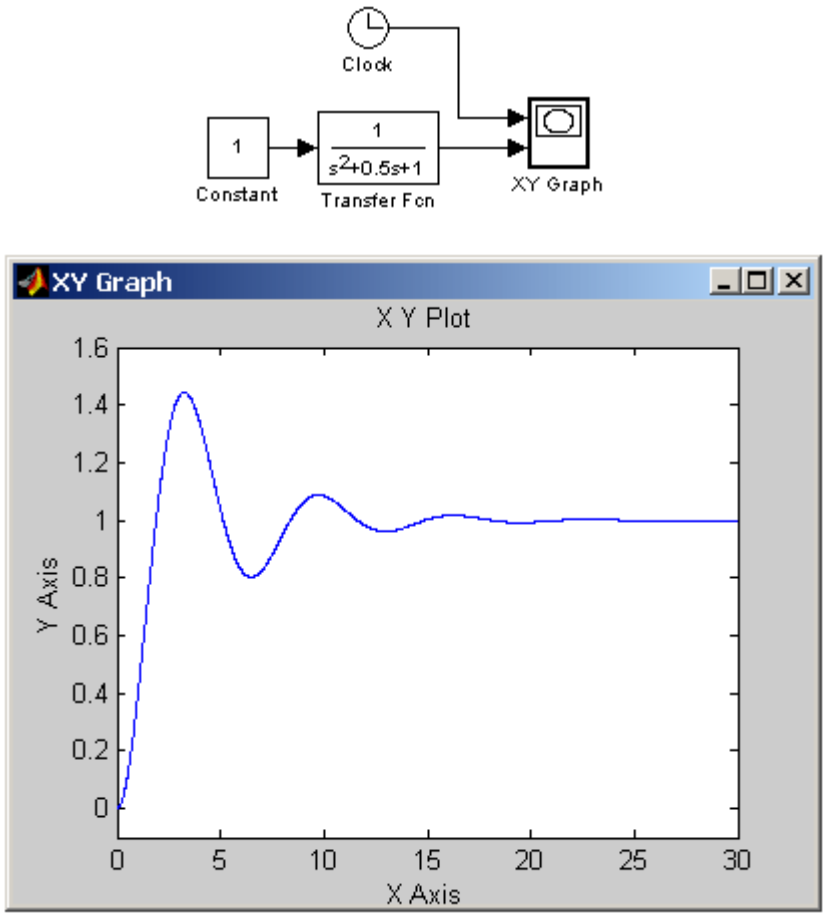


Рис. 3.15.

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

Графопостроитель можно использовать и для построения временных зависимостей. Для этого на первый вход следует подать временной сигнал с выхода блока **Clock**.

4. Цифровой дисплей Display.

Назначение:

Отображает значение сигнала в виде числа.

Параметры:

1. **Format** – формат отображения данных. Параметр **Format** может принимать следующие значения:
   - **short** – 5 значащих десятичных цифр.
   - **long** – 15 значащих десятичных цифр.
   - **short_e** – 5 значащих десятичных цифр и 3 символа степени десяти.
   - **long_e** – 15 значащих десятичных цифр и 3 символа степени десяти.
   - **bank** – «денежный» формат. Формат с фиксированной точкой и двумя десятичными цифрами в дробной части числа.
2. **Decimation** – кратность отображения входного сигнала. При **Decimation=1** отображается каждое значение входного сигнала, при **Decimation=2** отображается каждое второе значение и т. д.
3. **Sample time** – шаг модельного времени. Определяет дискретность отображения данных.
4. **Floating display** (флажок)– перевод блока в «свободный» режим. В данном режиме входной порт блока отсутствует, а выбор сигнала для отображения выполняется щелчком левой клавиши «мыши» на соответствующей лини связи.

5. Блок остановки моделирования Stop Simulation.

Назначение:

Обеспечивает завершение расчета, если входной сигнал блока становится не равным нулю.

Параметры:

Нет.

При подаче на вход блока ненулевого сигнала **Simulink** выполняет текущий шаг расчета, а затем останавливает моделирование. Если на вход блока подан векторный сигнал, то для остановки расчета достаточно, чтобы один элемент вектора стал ненулевым.

6. Блок сохранения данных в файле To File.

Назначение:

Блок записывает данные, поступающие на его вход, в файл.

Параметры:

1. **Filename** – имя файла для записи.
2. **Variable name** – имя переменной, содержащей записываемые данные.
3. **Decimation** – кратность записи в файл входного сигнала.
4. **Sample time** – шаг модельного времени. Определяет дискретность записи данных.

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

7. Блок сохранения данных в рабочей области To Workspace.

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

Параметры:

1. **Variable name** – имя переменной, содержащей записываемые данные.
2. **Limit data points to last** – максимальное количество сохраняемых расчетных точек по времени. В том случае, если значение параметра **Limit data points to last** задано как **inf**, то в рабочей области будут сохранены все данные.
3. **Decimation** – кратность записи данных в рабочую область.
4. **Sample time** – шаг модельного времени. Определяет дискретность записи данных.
5. **Save format** – формат сохранения данных. Может принимать значения:
6. **Matrix** – матрица. Данные сохраняются как массив, в котором число строк определяется числом расчетных точек по времени, а число столбцов – размерностью вектора подаваемого на вход блока. Если на вход подается скалярный сигнал, то матрица будет содержать лишь один столбец.
7. **Structure** – структура. Данные сохраняются в виде структуры, имеющей три поля: **time** – время, **signals** – сохраняемые значения сигналов, **blockName** – имя модели. Поле **time** для данного формата остается не заполненным.
8. **Structure with Time** – структура с дополнительным полем (время). Для данного формата, в отличие от предыдущего, поле **time** заполняется значениями времени.

8. Концевой приемник Terminator.

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

Параметры: Нет.

В том случае, если выход блока оказывается не подключенным ко входу другого блока, **Simulink** выдает предупреждающее сообщение в командном окне **MATLAB**. Для исключения этого необходимо использовать блок **Terminator**.

9. Блок выходного порта Outport.

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

Параметры:

1. **Port number** – номер порта.
2. **Output when disabled** – вид сигнала на выходе подсистемы в случае, если подсистема выключена. Может принимать значения:
3. **held** – выходной сигнал подсистемы равен последнему рассчитанному значению.

4. **reset** – выходной сигнал подсистемы равен значению, задаваемому параметром **Initial output**.
5. **Initial output** – значение сигнала на выходе подсистемы до начала ее работы и в случае, если подсистема выключена. Используется для управляемых подсистем.

3.6. Continuous – аналоговые блоки

1. Блок вычисления производной Derivative.

Назначение:

Выполняет численное дифференцирование входного сигнала.

Параметры:

Нет.

Для вычисления производной используется приближенная формула Эйлера.

Точность вычисления производной существенно зависит от величины установленного шага расчета. Выбор меньшего шага расчета улучшает точность вычисления производной.

2. Интегрирующий блок Integrator.

Назначение:

Выполняет интегрирование входного сигнала.

Параметры:

1. **External reset** – Внешний сброс. Тип внешнего управляющего сигнала, обеспечивающего сброс интегратора к начальному состоянию. Выбирается из списка:
   - **none** – нет (сброс не выполняется);
   - **rising** – нарастающий сигнал (передний фронт сигнала);
   - **falling** – спадающий сигнал (задний фронт сигнала);
   - **either** – нарастающий либо спадающий сигнал;
   - **level** – не нулевой сигнал (сброс выполняется, если сигнал на управляющем входе становится не равным нулю).

В том случае, если выбран какой-либо (но не **none**) тип управляющего сигнала, то на изображении блока появляется дополнительный управляющий вход. Рядом с дополнительным входом будет показано условное обозначение управляющего сигнала.

2. **Initial condition source** – Источник начального значения выходного сигнала. Выбирается из списка:
   - **internal** – внутренний;
   - **external** – внешний. В этом случае на изображении блока появляется дополнительный вход, обозначенный x0, на который необходимо подать сигнал, задающий начальное значение выходного сигнала интегратора.

3. **Initial condition** – Начальное условие. Установка начального значения выходного сигнала интегратора. Параметр доступен, если выбран внутренний источник начального значения выходного сигнала.
4. **Limit output** (флажок) – Использование ограничения выходного сигнала.
5. **Upper saturation limit** – Верхний уровень ограничения выходного сигнала. Может быть задан как числом, так и символьной последовательностью **inf**, то есть +∞.
6. **Lower saturation limit** – Нижний уровень ограничения выходного сигнала. Может быть задан как числом, так и символьной последовательностью **inf**, то есть **-**∞.
7. **Show saturation port** – управляет отображением порта, выводящего сигнал, свидетельствующий о выходе интегратора на ограничение. Выходной сигнал данного порта может принимать следующие значения:
   - **Ноль**, если интегратор не находится на ограничении.
   - **+1**, если выходной сигнал интегратора достиг верхнего ограничивающего предела.
   - **-1**, если выходной сигнал интегратора достиг нижнего ограничивающего предела.
8. **Show state port** (флажок) – Отобразить / скрыть порт состояния блока. Данный порт используется в том случае, если выходной сигнал интегратора требуется подать в качестве сигнала обратной связи этого же интегратора. Например, при установке начальных условий через внешний порт или при сбросе интегратора через порт сброса. Выходной сигнал с этого порта может использоваться также для организации взаимодействия с управляемой подсистемой.
9. **Absolute tolerance** – Абсолютная погрешность.

3. Блок Memory.

Назначение:

Выполняет задержку входного сигнала на один временной такт.

Параметры:

1. **Initial condition** – начальное значение выходного сигнала.
2. **Inherit sample time** (флажок) – Наследовать шаг модельного времени. Если этот флажок установлен, то блок **Memory** использует шаг модельного времени (**Sample time**) такой же, как и в предшествующем блоке.

4. Блок фиксированной задержки сигнала Transport Delay.

Назначение:

Обеспечивает задержку входного сигнала на заданное время.

Параметры:

1. **Time Delay** – Время задержки сигнала (не отрицательное значение).
2. **Initial input** – Начальное значение выходного сигнала.

3. **Buffer size** – Размер памяти, выделяемой для хранения задержанного сигнала. Задается в байтах числом, кратным 8 (по умолчанию 1024).
4. **Pade order (for linearization)** – Порядок ряда Паде, используемого при аппроксимации выходного сигнала. Задается целым положительным числом.

При выполнении моделирования значение сигнала и соответствующее ему модельное время сохраняются во внутреннем буфере блока **Transport Delay**. По истечении времени задержки значение сигнала извлекается из буфера и передается на выход блока. В том случае, если шаги модельного времени не совпадают со значениями моментов времени для записанного в буфер сигнала, блок **Transport Delay** выполняет аппроксимацию выходного сигнала.

5. Блок управляемой задержки сигнала Variable Transport Delay.

Назначение:

Выполняет задержку входного сигнала, заданную величиной сигнала управления.

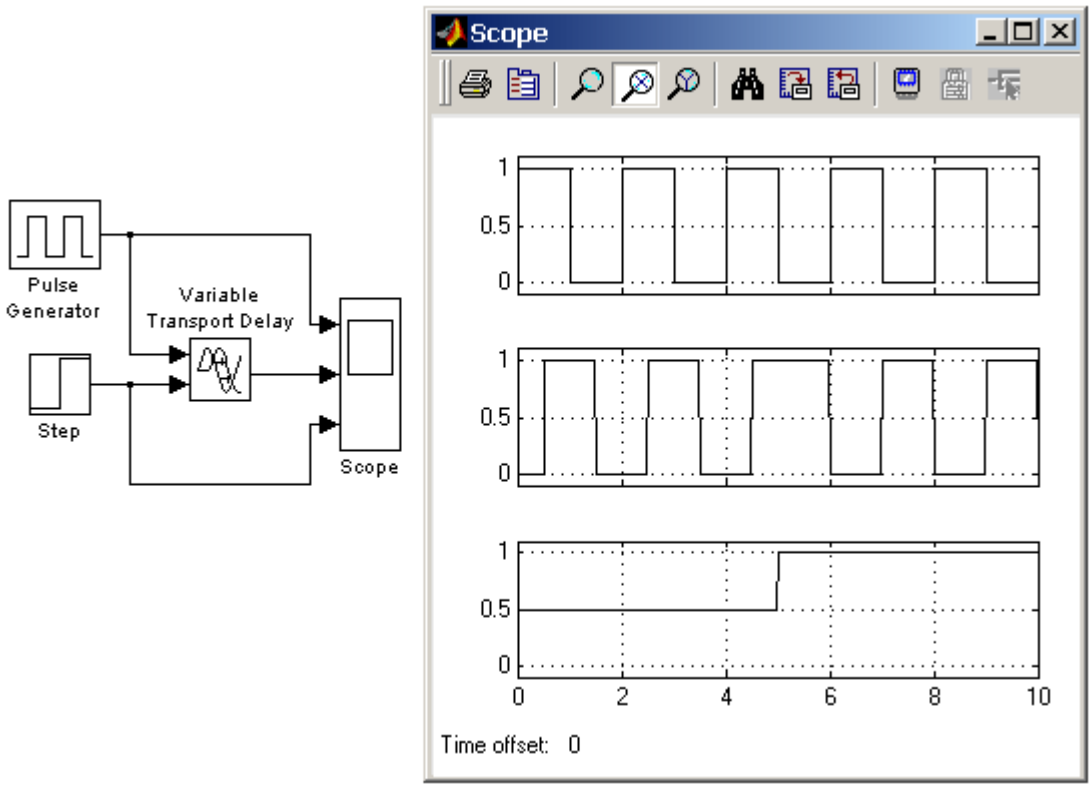


Рис. 3.16.

Параметры:

1. **Maximum delay** – Максимальное значение времени задержки сигнала (не отрицательное значение).
2. **Initial input** – Начальное значение выходного сигнала.
3. **Buffer size** – Размер памяти, выделяемой для хранения задержанного сигнала. Задается в байтах числом, кратным 8 (по умолчанию 1024).

4. **Pade order (for linearization)** – Порядок ряда Паде, используемого при аппроксимации выходного сигнала. Задается целым положительным числом.

Блок управляемой задержки **Variable Transport Delay** работает аналогично блоку постоянной задержки сигнала **Transport Delay**.

В том случае, если значение управляющего сигнала, задающего величину задержки, превышает значение, заданное параметром **Maximum delay**, то задержка выполняется на величину **Maximum delay**.

6. Блок передаточной функции Transfer Fcn.

Назначение:

Блок передаточной характеристики **Transfer Fcn** задает передаточную функцию в виде отношения полиномов:

$$H(s)=\frac{y(s)}{u(s)}=\frac{num(s)}{den(s)}=\frac{num(1)s^{nn-1}+num(2)s^{nn-2}+...+num(nn)}{den(1)s^{nd-1}+den(2)s^{nd-2}+...+den(nd)}$$, где

nn и **nd** – порядок числителя и знаменателя передаточной функции;

num – вектор или матрица коэффициентов числителя;

den – вектор коэффициентов знаменателя.

Параметры:

1. **Numerator** – вектор или матрица коэффициентов полинома числителя.
2. **Denominator** – вектор коэффициентов полинома знаменателя.
3. **Absolute tolerance** – Абсолютная погрешность.

Порядок числителя не должен превышать порядок знаменателя.

Входной сигнал блока должен быть скалярным. В том случае, если коэффициенты числителя заданы вектором, то выходной сигнал блока будет также скалярным (как и входной сигнал). Начальные условия при использовании блока **Transfer Fcn** полагаются нулевыми.

7. Блок передаточной функции Zero-Pole.

Назначение:

Блок **Zero-Pole** определяет передаточную функцию с заданными полюсами и нулями

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

Z – вектор или матрица нулей передаточной функции (корней полинома числителя);

P – вектор полюсов передаточной функции (корней полинома знаменателя);

K – коэффициент передаточной функции, или вектор коэффициентов, если нули передаточной функции заданы матрицей. При этом размерность вектора **K** определяется числом строк матрицы нулей.

Параметры:

1. **Zeros** – Вектор или матрица нулей.
2. **Poles** – Вектор полюсов.

3. **Gain** – Скалярный или векторный коэффициент передаточной функции.

4. **Absolute tolerance** – Абсолютная погрешность.

Количество нулей не должно превышать число полюсов передаточной функции. В том случае, если нули передаточной функции заданы матрицей, то блок **Zero-Pole** моделирует векторную передаточную функцию.

Нули или полюса могут быть заданы комплексными числами. В этом случае нули или полюса должны быть заданы комплексно-сопряженными парами полюсов или нулей, соответственно. Начальные условия при использовании блока **Zero-Pole** полагаются нулевыми.

8. Блок модели динамического объекта State-Space.

Назначение:

Блок создает динамический объект, описываемый уравнениями в пространстве состояний

$$x = Ax + Bu;$$

$$y = Cx + Du, \text{ где}$$

x – вектор состояния;

u – вектор входных воздействий;

y – вектор выходных сигналов;

A, B, C, D – матрицы: системы, входа, выхода и обхода, соответственно.

Размерность матриц показана на рисунке 3.17 (**n** – количество переменных состояния, **m** – число входных сигналов, **r** – число выходных сигналов).

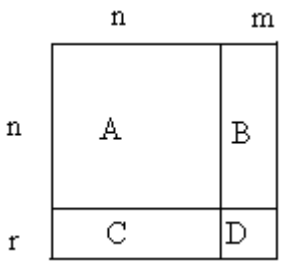


Рис. 3.17.

Параметры:

1. **A** – Матрица системы.
2. **B** – Матрица входа.
3. **C** – Матрица выхода.
4. **D** – Матрица обхода.
5. **Initial condition** – Вектор начальных условий.
6. **Absolute tolerance** – Абсолютная погрешность.

3.7. Discrete – дискретные блоки

1. Блок единичной дискретной задержки Unit Delay.

Назначение:

Выполняет задержку входного сигнала на один шаг модельного времени.

Параметры:

1. **Initial condition** – Начальное значение для выходного сигнала.
2. **Sample time** – Шаг модельного времени.

Входной сигнал блока может быть как скалярным, так и векторным. При векторном входном сигнале задержка выполняется для каждого элемента вектора. Блок поддерживает работу с комплексными и действительными сигналами.

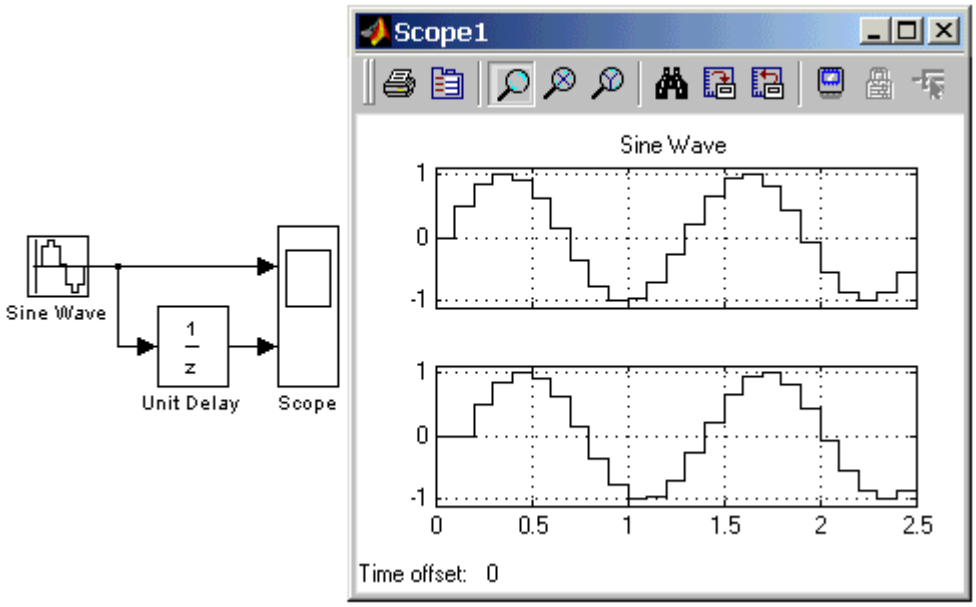


Рис. 3.18.

2. Блок экстраполятора нулевого порядка Zero-Order Hold.

Назначение:

Блок выполняет дискретизацию входного сигнала по времени.

Параметры:

1. **Sample time** – Величина шага дискретизации по времени.

Блок фиксирует значение входного сигнала в начале интервала квантования и поддерживает на выходе это значение до окончания интервала квантования. Затем выходной сигнал изменяется скачком до величины входного сигнала на следующем шаге квантования.

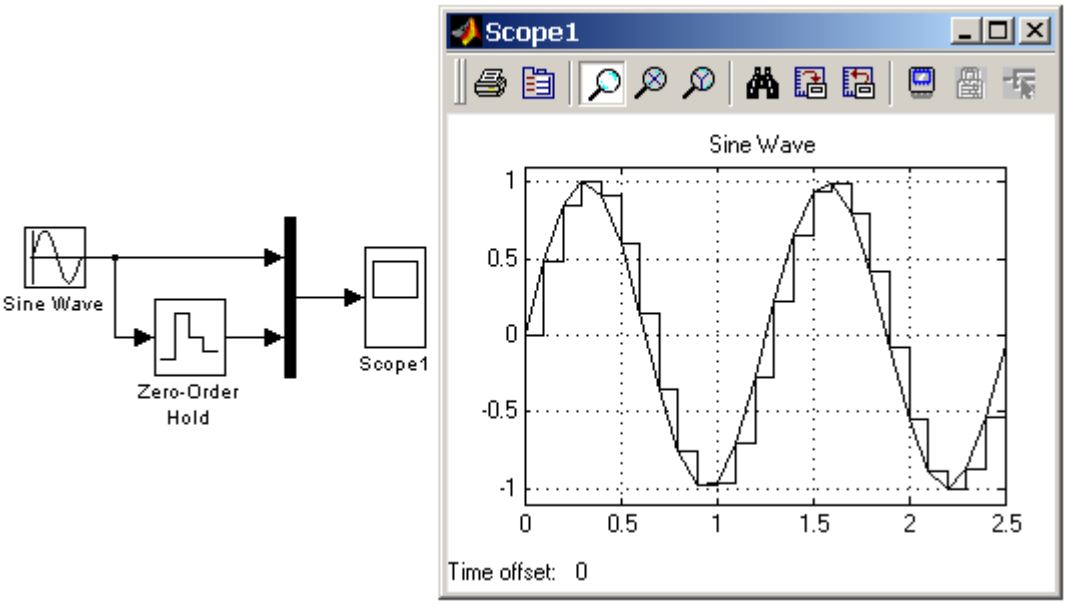


Рис. 3.19.

3. Блок экстраполятора первого порядка First-Order Hold.

Назначение:

Блок задает линейное изменение выходного сигнала на каждом такте дискретизации, в соответствии с крутизной входного сигнала на предыдущем интервале дискретизации.

Параметры:

1. **Sample time** – Величина шага дискретизации по времени.

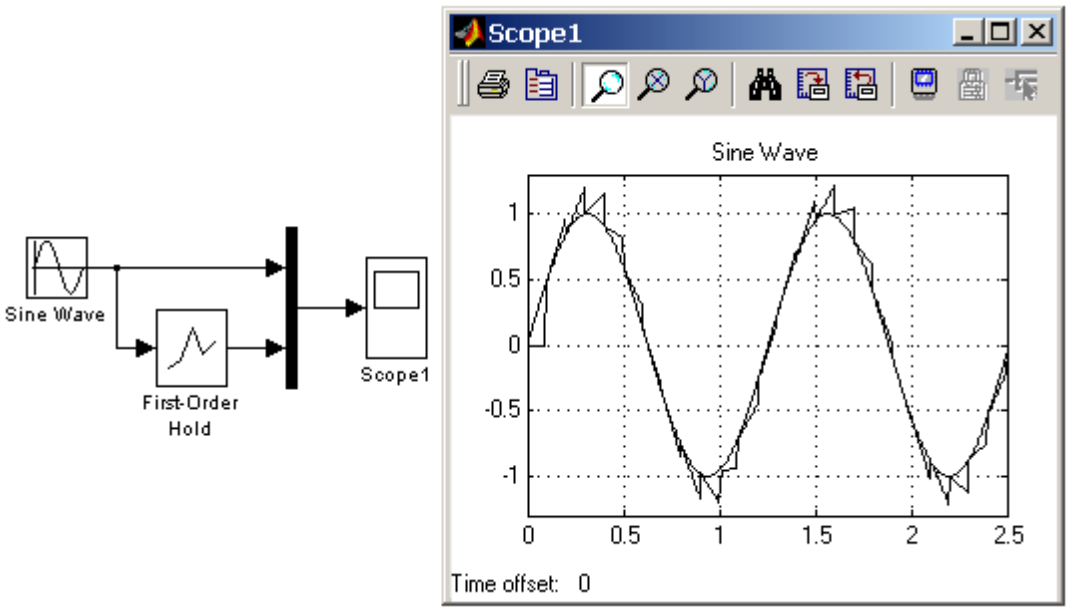


Рис. 3.20.

4. Блок дискретного интегратора Discrete-Time Integrator.

Назначение:

Блок используется для выполнения операции интегрирования в дискретных системах.

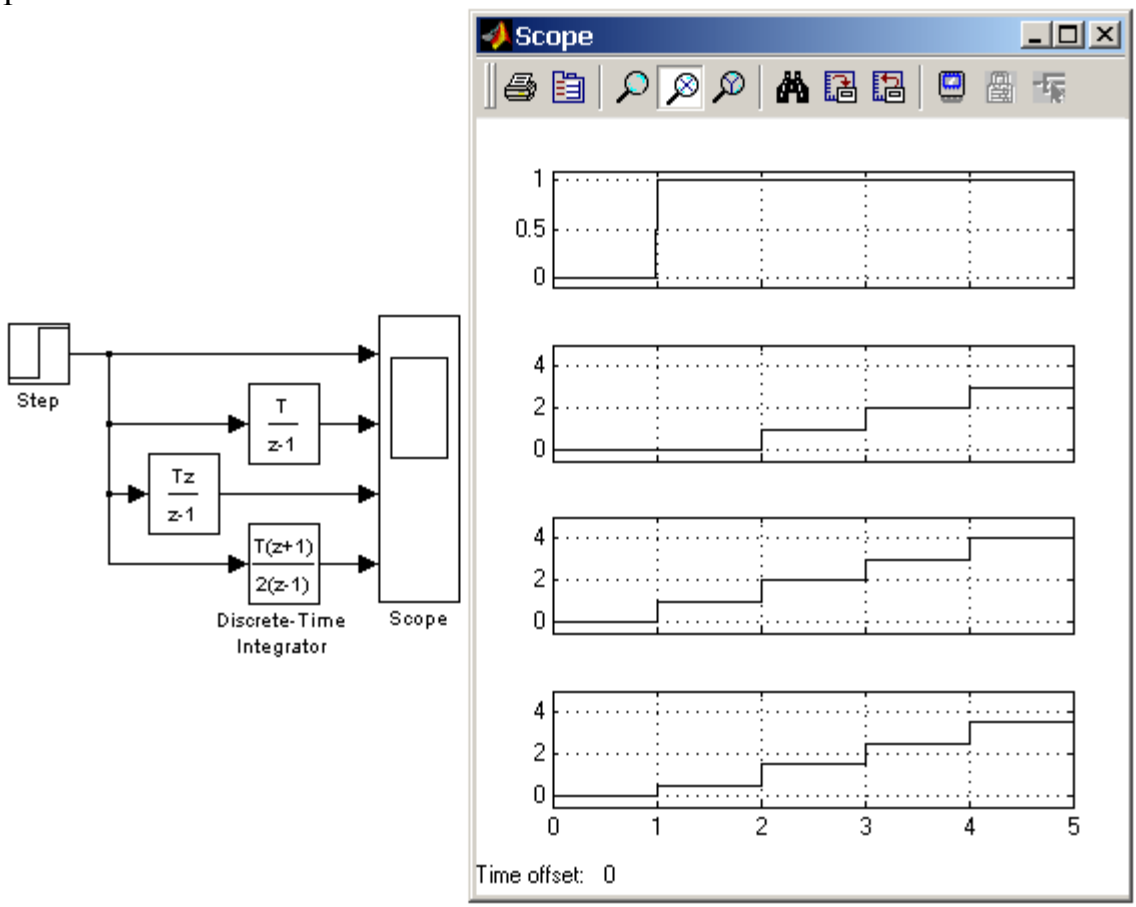


Рис. 3.21.

Параметры:

1. **Integration method** – Метод численного интегрирования:

- **Forward Euler** – Прямой метод Эйлера.

Метод использует аппроксимацию T/(z-1) передаточной функции 1/s. Выходной сигнал блока рассчитывается по выражению

$$y(k) = y(k–1) + T \cdot u(k–1).$$

y – выходной сигнал интегратора;
u – входной сигнал интегратора;
T – шаг дискретизации;
k – номер шага моделирования.

- **Backward Euler** – Обратный метод Эйлера.

Метод использует аппроксимацию $T \cdot z /(z–1)$ передаточной функции 1/s. Выходной сигнал блока рассчитывается по выражению

$$y(k) = y(k–1) + T \cdot u(k).$$

- **Trapeziodal** – Метод трапеций.

Метод использует аппроксимацию T/2·(z+1)/(z–1) передаточной функции 1/s. Выходной сигнал блока рассчитывается по выражению x(k) = y(k–1) + T/2 · u(k–1).

2. **Sample time** – Шаг дискретизации по времени.

Остальные параметры дискретного интегратора те же, что и у блока аналогового интегратора **Integrator** (библиотека **Continuous**).

5. Дискретная передаточная функция Discrete Transfer Fcn.

Назначение:

Блок Discrete Transfer Fcn задает дискретную передаточную функцию в виде отношения полиномов:

$$H(z) = \frac{num(z)}{den(z)} = \frac{num_0 z^n + num_1 z^{n-1} + \ldots + num_m z^{n-m}}{den_0 z^n + den_1 z^{n-1} + \ldots + den_n}, \text{ где}$$

m+1 и n+1 – количество коэффициентов числителя и знаменателя, соответственно;

num – вектор или матрица коэффициентов числителя;

den – вектор коэффициентов знаменателя.

Параметры:

1. **Numerator** – Вектор или матрица коэффициентов числителя.
2. **Denominator** – Вектор коэффициентов знаменателя.
3. **Sample time** – Шаг дискретизации по времени.

Порядок числителя не должен превышать порядок знаменателя.

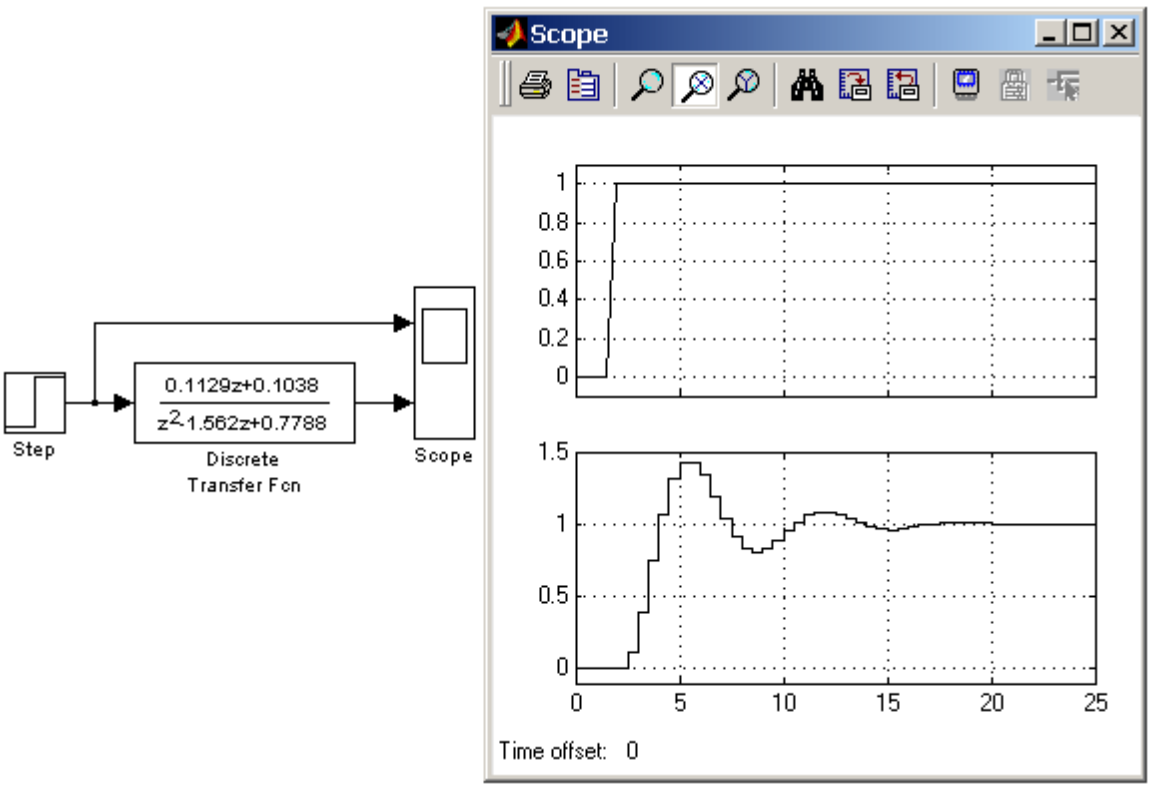


Рис. 3.22.

6. Блок дискретной передаточной функции Discrete Zero-Pole.

Назначение:

Блок Discrete Zero-Pole определяет дискретную передаточную функцию с заданными полюсами и нулями:

$$H(z) = K\frac{Z(z)}{P(z)} = K\frac{(z-Z_1)(z-Z_2)...(z-Z_m)}{(z-P_1)(z-P_2)...(z-P_n)}, \text{ где}$$

Z – вектор или матрица нулей передаточной функции;

P – вектор полюсов передаточной функции;

K – коэффициент передаточной функции, или вектор коэффициентов, если нули передаточной функции заданы матрицей. При этом размерность вектора K определяется числом строк матрицы нулей.

Параметры:

1. **Zeros** – Вектор или матрица нулей.
2. **Poles** – Вектор полюсов.
3. **Gain** – Скалярный или векторный коэффициент передаточной функции.
4. **Sample time** – Шаг дискретизации по времени.

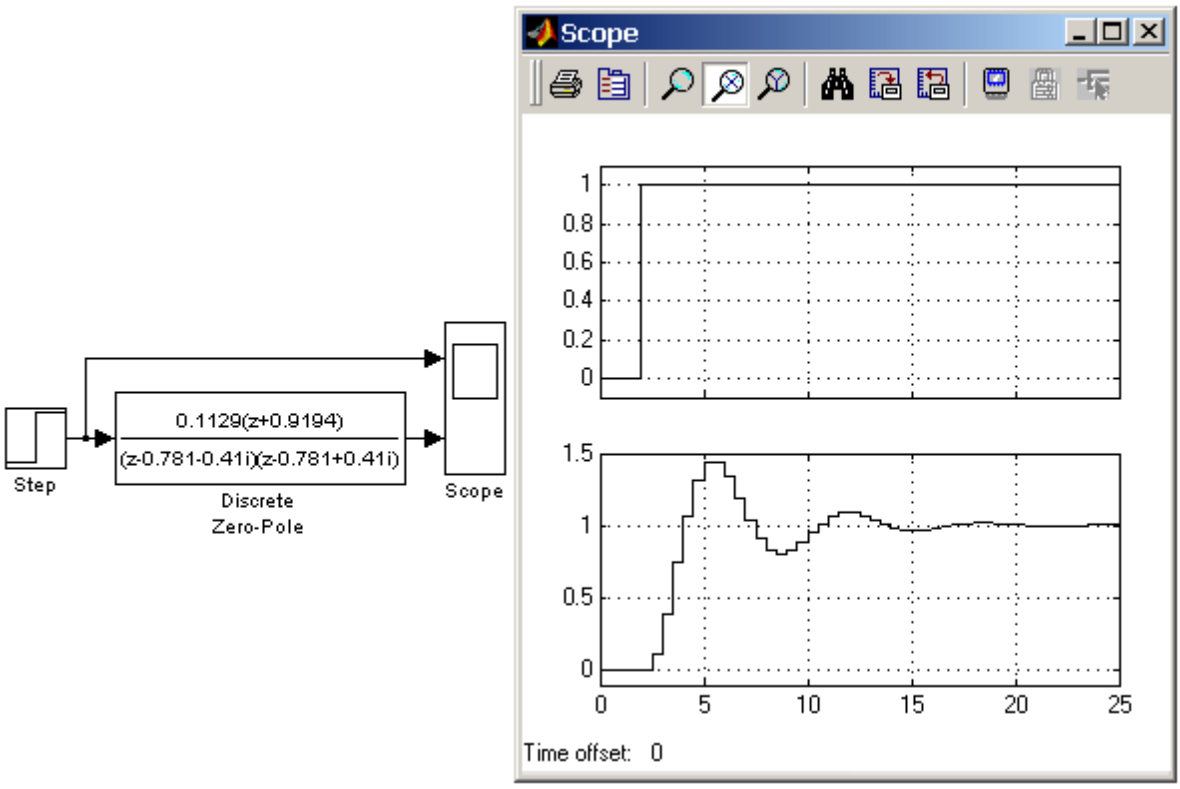


Рис. 3.23.

Количество нулей не должно превышать число полюсов передаточной функции.

В том случае, если нули передаточной функции заданы матрицей, то блок **Discrete Zero-Pole** моделирует векторную передаточную функцию.

Нули или полюса могут быть заданы комплексными числами. В этом случае нули или полюса должны быть заданы комплексно-сопряженными парами полюсов или нулей, соответственно.

Начальные условия при использовании блока **Discrete Zero-Pole** полагаются нулевыми.

7. Блок дискретного фильтра Discrete Filter.

Назначение:

Блок дискретного фильтра **Discrete Filter** задает дискретную передаточную функцию от обратного аргумента (1/z)

$$H(1/z)=\frac{num(1/z)}{den(1/z)}=\frac{num_0z^0+num_1z^{-1}+num_2z^{-2}+...+num_mz^{-m}}{den_0z^0+den_1z^{-1}+den_2z^{-2}+...+den_mz^{-n}}$$, где

m+1 и n+1 – количество коэффициентов числителя и знаменателя, соответственно;

num – вектор или матрица коэффициентов числителя;

den – вектор коэффициентов знаменателя.

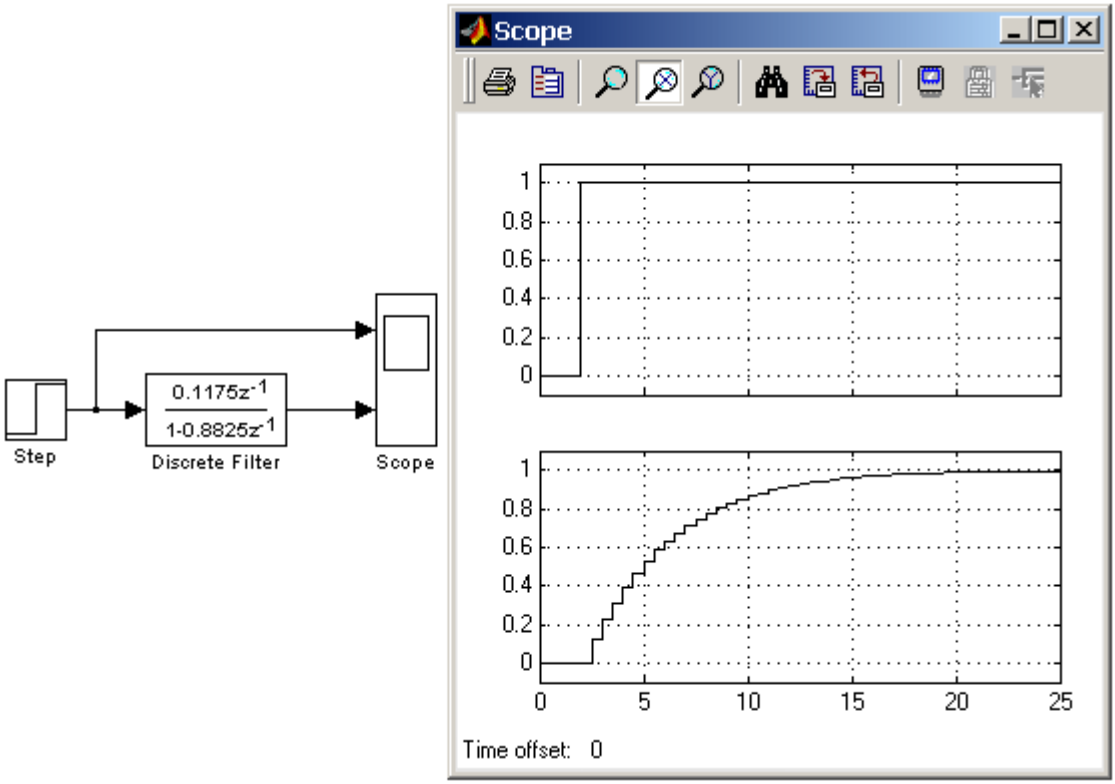


Рис. 3.24.

Параметры:

1. **Numerator** – Вектор или матрица коэффициентов числителя.
2. **Denominator** –Вектор коэффициентов знаменателя.
3. **Sample time** – Шаг дискретизации по времени.

8. Блок модели динамического объекта Discrete State-Space.

Назначение:

Блок создает динамический объект, описываемый уравнениями в пространстве состояний

$$x(n+1) = Ax(n) + Bu(n);$$
$$y(n) = Cx(n) + Du(n), \text{ где}$$

x – вектор состояния;

u – вектор входных воздействий;

y – вектор выходных сигналов;

A, B, C, D – матрицы: системы, входа, выхода и обхода, соответственно;

n – номер шага моделирования.

Размерность матриц показана на рисунке 3.25 (n – количество переменных состояния, m – число входных сигналов, r – число выходных сигналов).

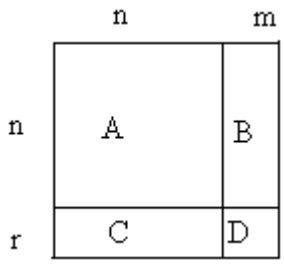


Рис. 3.25.

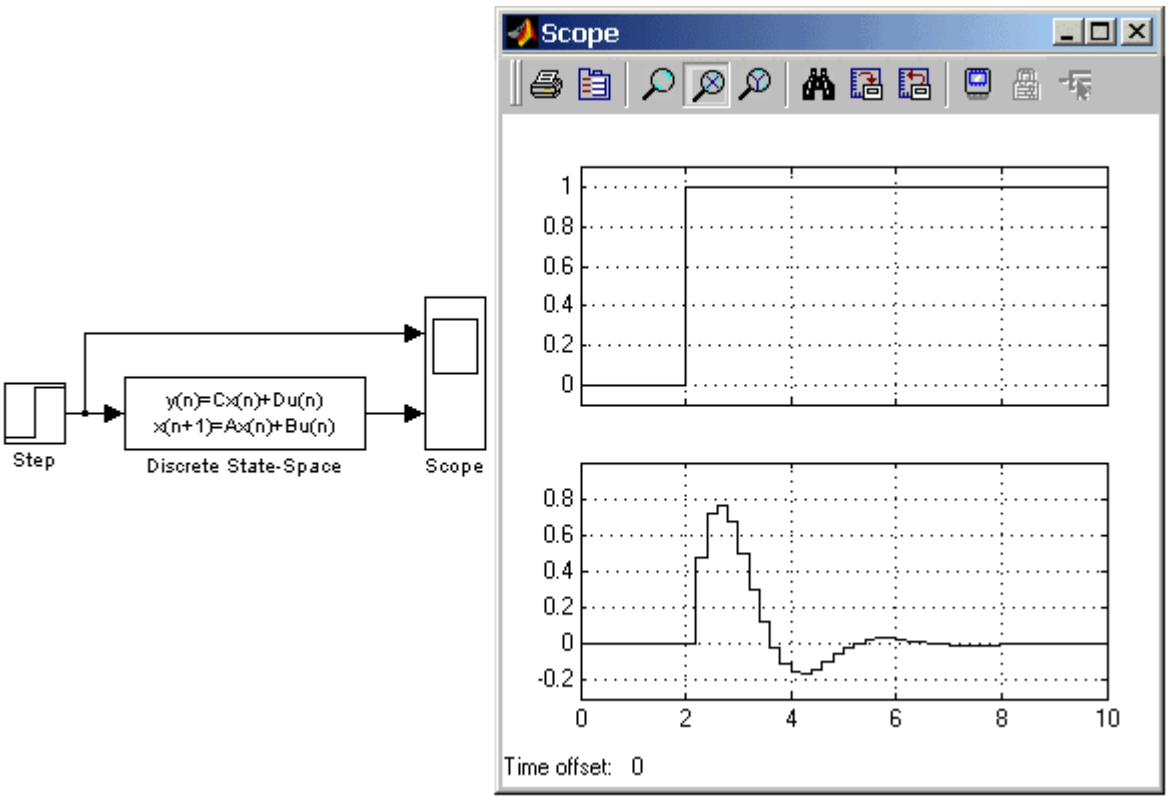


Рис. 3.26.

Параметры:

1. **A** – Матрица системы.
2. **B** – Матрица входа.
3. **C** – Матрица выхода.
4. **D** – Матрица обхода.
5. **Initial condition** – Вектор начальных условий.
6. **Sample time** – Шаг дискретизации по времени.

3.8. Nonlinear – нелинейные блоки

1. Блок ограничения Saturation.

Назначение:

Выполняет ограничение величины сигнала.

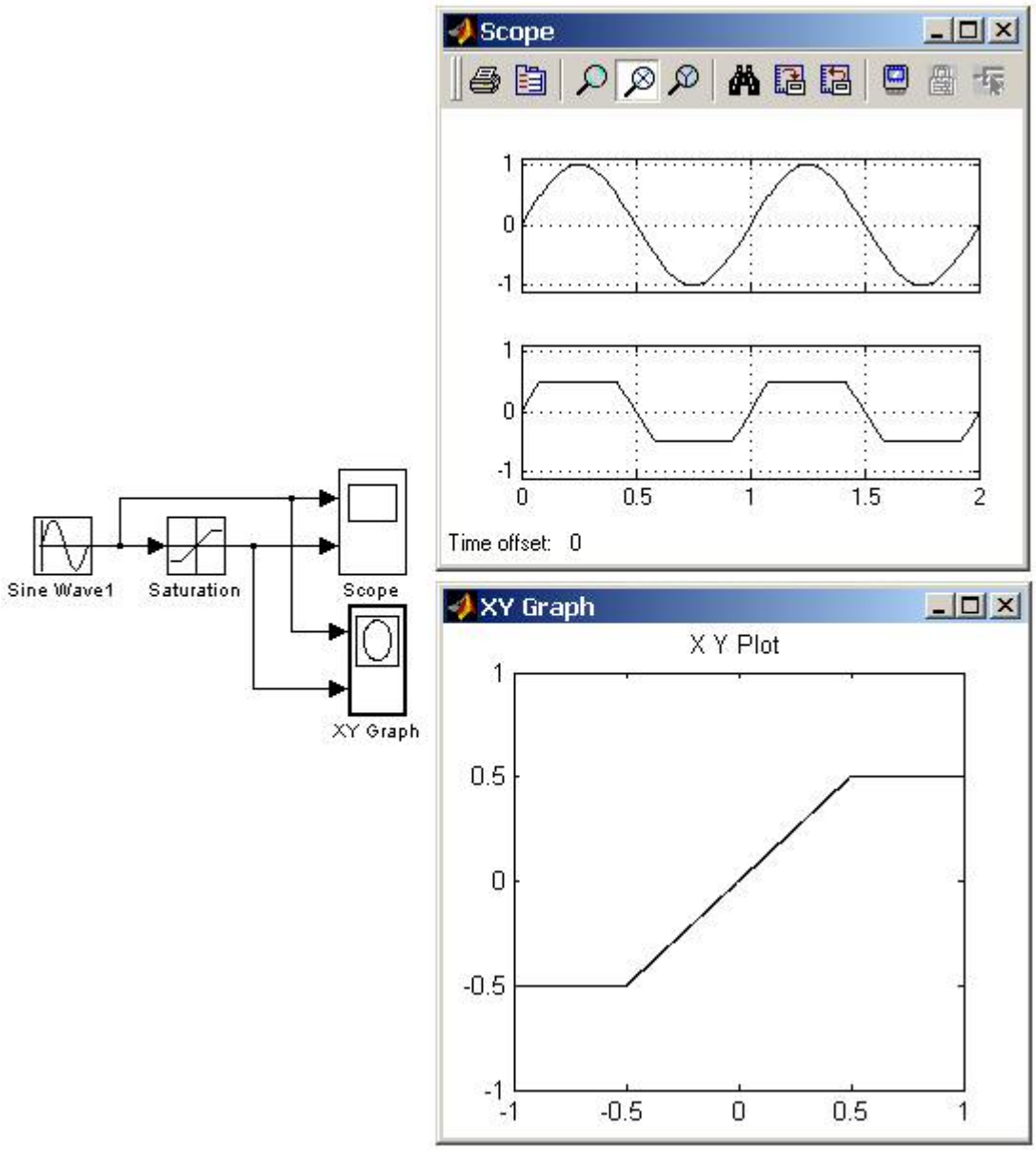


Рис. 3.27.

Параметры:

1. **Upper limit** – Верхний порог ограничения.
2. **Lower limit** – Нижний порог ограничения.
3. **Treat as gain when linearizing** (флажок) – Трактовать как усилитель с коэффициентом передачи равным 1 при линеаризации.

Выходной сигнал блока равен входному, если его величина не выходит за порог ограничения. По достижении входным сигналом уровня ограничения выходной сигнал блока перестает изменяться и остается равным порогу.

2. Блок с зоной нечувствительности Dead Zone.

Назначение: Реализует зависимость типа «зона нечувствительности».

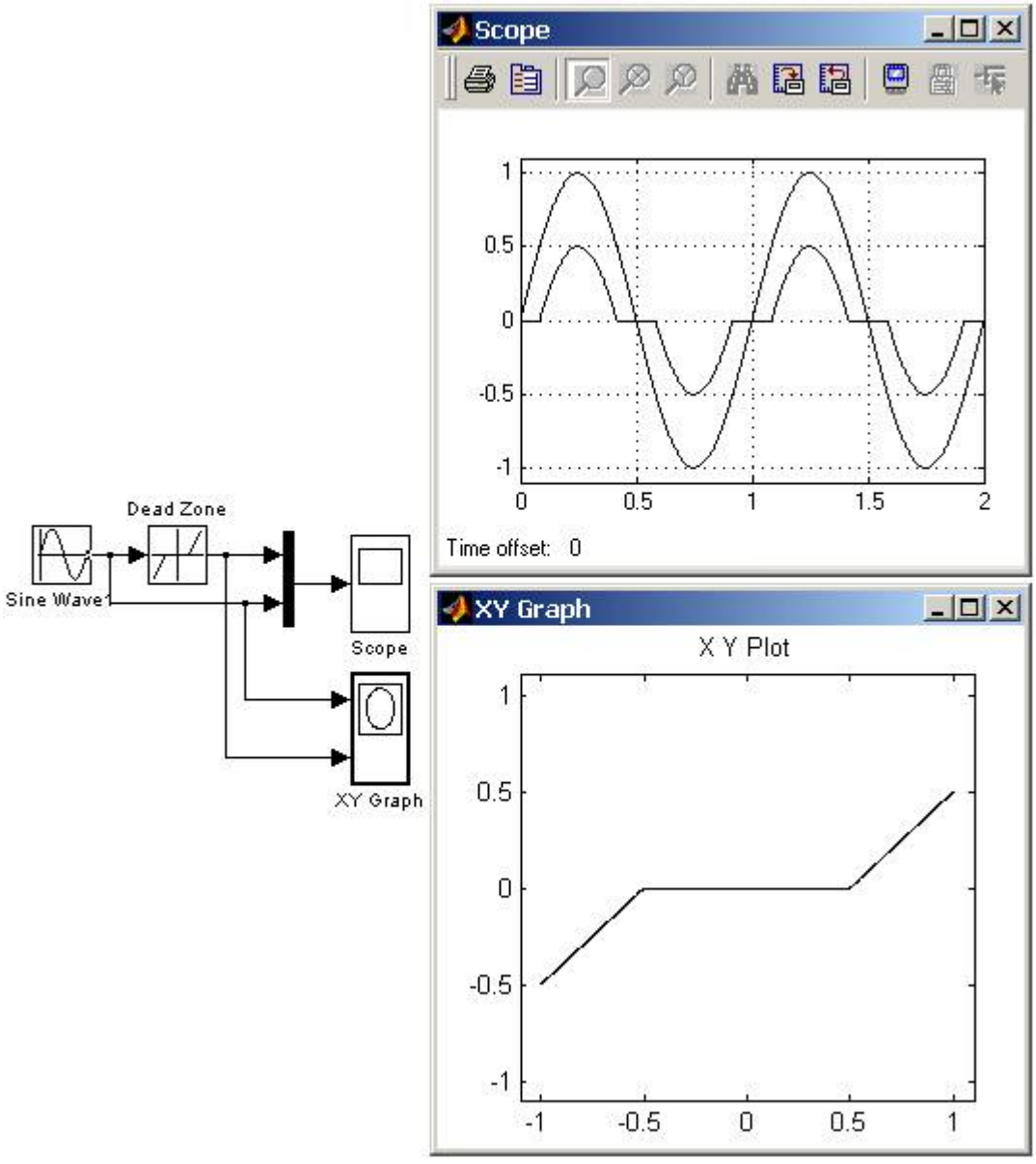


Рис. 3.28.

Выходной сигнал блока вычисляется в соответствии со следующим алгоритмом:

- Если величина входного сигнала находится в пределах зоны нечувствительности, то выходной сигнал блока равен нулю.
- Если входной >= верхнему входному порогу зоны нечувствительности, то выходной сигнал равен входному минус величина порога.
- Если входной сигнал <= нижнему входному порогу зоны нечувствительности, то выходной сигнал равен входному минус величина порога.

Параметры:

1. **Start of dead zone** – Начало зоны нечувствительности (нижний порог).
2. **End of dead zone** – Конец зоны нечувствительности (верхний порог).
3. **Saturate on integer overflow** (флаг) – Подавлять переполнение целого. Если установлено – ограничение сигналов целого типа выполняется корректно.
4. **Treat as gain when linearizing** (флажок) – Трактовать как усилитель с коэффициентом передачи равным 1 при линеаризации.

3. Релейный блок Relay.

Назначение:

Реализует релейную нелинейность.

Параметры:

1. **Switch on point** – Порог включения. Значение, при котором происходит включение реле.
2. **Switch off point** – Порог выключения. Значение, при котором происходит выключение реле.
3. **Output when on** – Величина выходного сигнала во включенном состоянии.
4. **Output when off** – Величина выходного сигнала в выключенном состоянии.

Выходной сигнал блока может принимать два значения. Одно из них соответствует включенному состоянию реле, второе – выключенному. Переход из одного состояния в другое происходит скачком при достижении входным сигналом порога включения или выключения реле. В том случае если пороги включения и выключения реле имеют разные значения, то блок реализует релейную характеристику с гистерезисом. При этом значение порога включения должно быть больше, чем значение порога выключения.

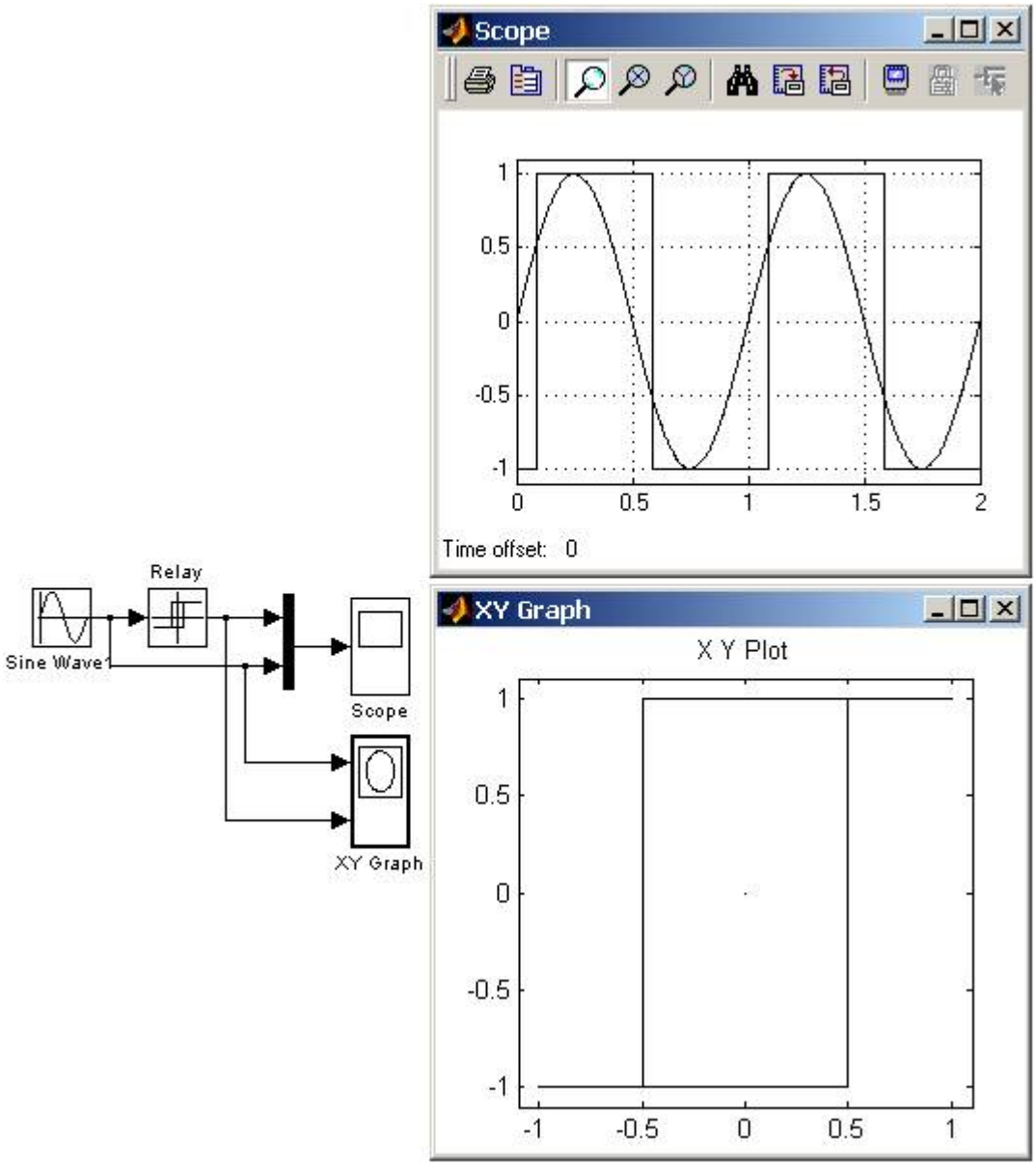


Рис. 3.29.

4. Блок ограничения скорости изменения сигнала Rate Limiter.

Назначение:

Блок обеспечивает ограничение скорости изменения сигнала (первой производной).

Параметры:

1. **Rising slew rate** – Уровень ограничения скорости при увеличении сигнала.
2. **Falling slew rate** – Уровень ограничения скорости при уменьшении сигнала.

Вычисление производной сигнала выполняется по выражению

$$rate = \frac{u(i) - y(i-1)}{t(i) - t(i-1)}, \text{ где}$$

u(i) – значение входного сигнала на текущем шаге;

t(i) – значение модельного времени на текущем шаге;

y(i-1) – значение выходного сигнала на предыдущем шаге;

t(i-1) – значение модельного времени на предыдущем шаге.

Вычисленное значение производной сравнивается со значениями уровней ограничения скорости **Rising slew rate** и **Falling slew rate**. Если значение производной больше, чем значение параметра **Rising slew rate**, то выходной сигнал блока вычисляется по выражению

$$y(i) = \Delta t \cdot R + y(i-1),$$

где R – уровень ограничения скорости при увеличении сигнала.

Если значение производной меньше, чем значение параметра **Falling slew rate**, то выходной сигнал блока вычисляется по выражению

$$y(i) = \Delta t \cdot F + y(i-1),$$

где F – уровень ограничения скорости при уменьшении сигнала.

Если значение производной лежит в пределах между нижним и верхним уровнями ограничения, то выходной сигнал блока равен входному

$$y(i) = u(i).$$

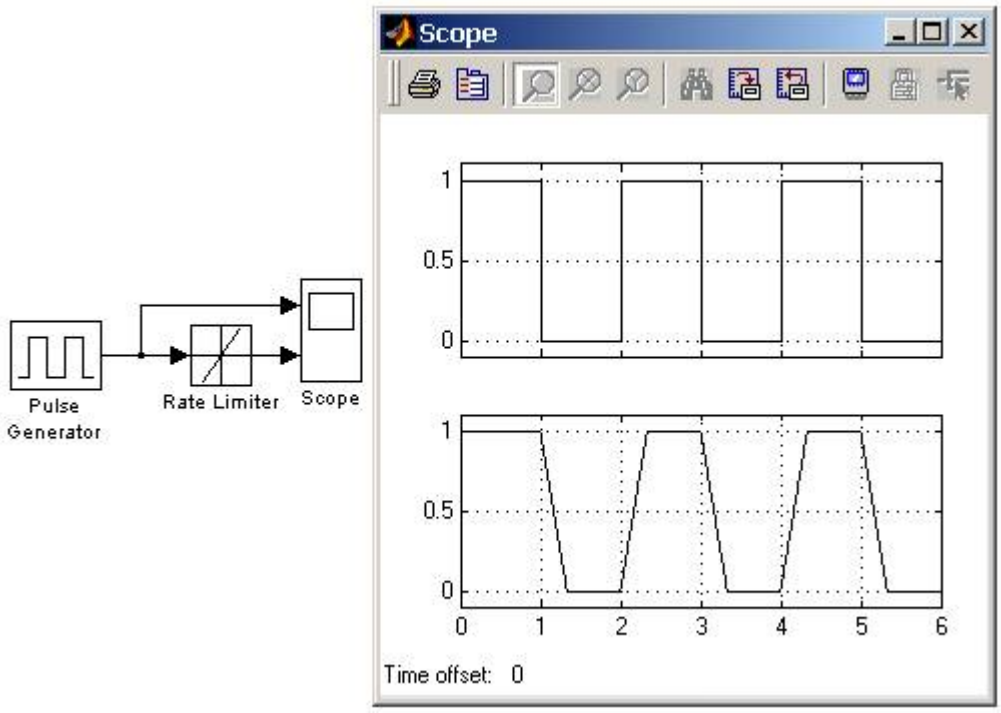


Рис. 3.30.

5. Блок квантования по уровню Quantizer.

Назначение: Блок обеспечивает квантование входного сигнала с одинаковым шагом по уровню.

Параметры: **Quantization interval** – шаг квантования по уровню.

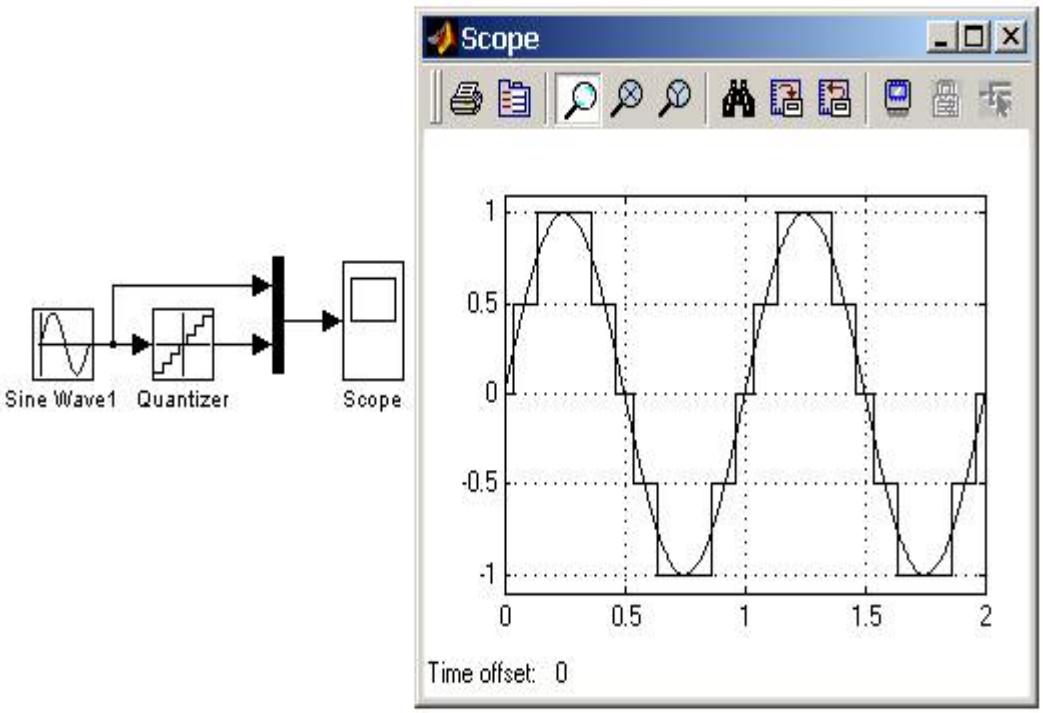


Рис. 3.31.

6. Блок переключателя Switch.

Назначение:

Выполняет переключение входных сигналов по сигналу управления.

Параметры: **Threshold** – Порог управляющего сигнала.

Блок работает следующим образом: Если сигнал управления, подаваемый на средний вход, меньше, чем величина порогового значения **Threshold**, то на выход блока проходит сигнал с первого (верхнего) входа. Если сигнал управления превысит пороговое значение, то на выход блока будет поступать сигнал со второго (нижнего) входа.

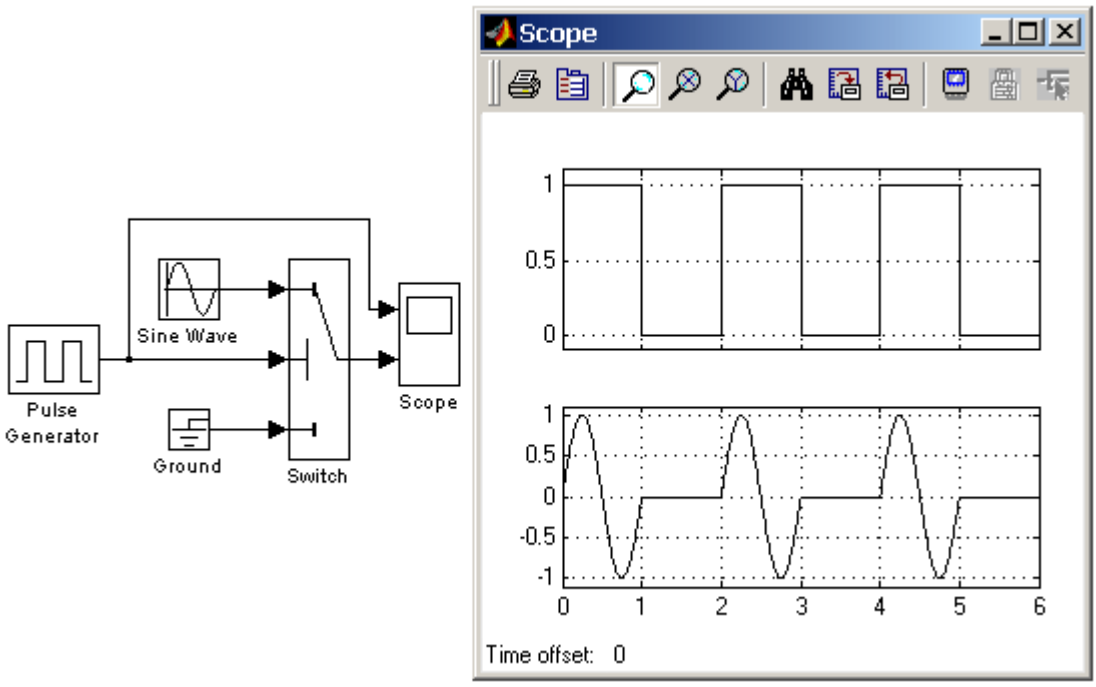


Рис. 3.32.

7. Блок сухого и вязкого трения Coulomb and Viscous Friction.

Назначение:

Моделирует эффекты сухого и вязкого трения.

Параметры:

1. **Coulomb friction value (Offset) –** Величина сухого трения.
2. **Coefficient of viscous friction (Gain) –** Коэффициент вязкого трения.

Блок реализует нелинейную характеристику, соответствующую выражению

$$y = sign(u) \cdot (Gain \cdot abs(u) + Offset), \text{ где}$$

u – входной сигнал;

y – выходной сигнал;

Gain – коэффициент вязкого трения;

Offset – Величина сухого трения.

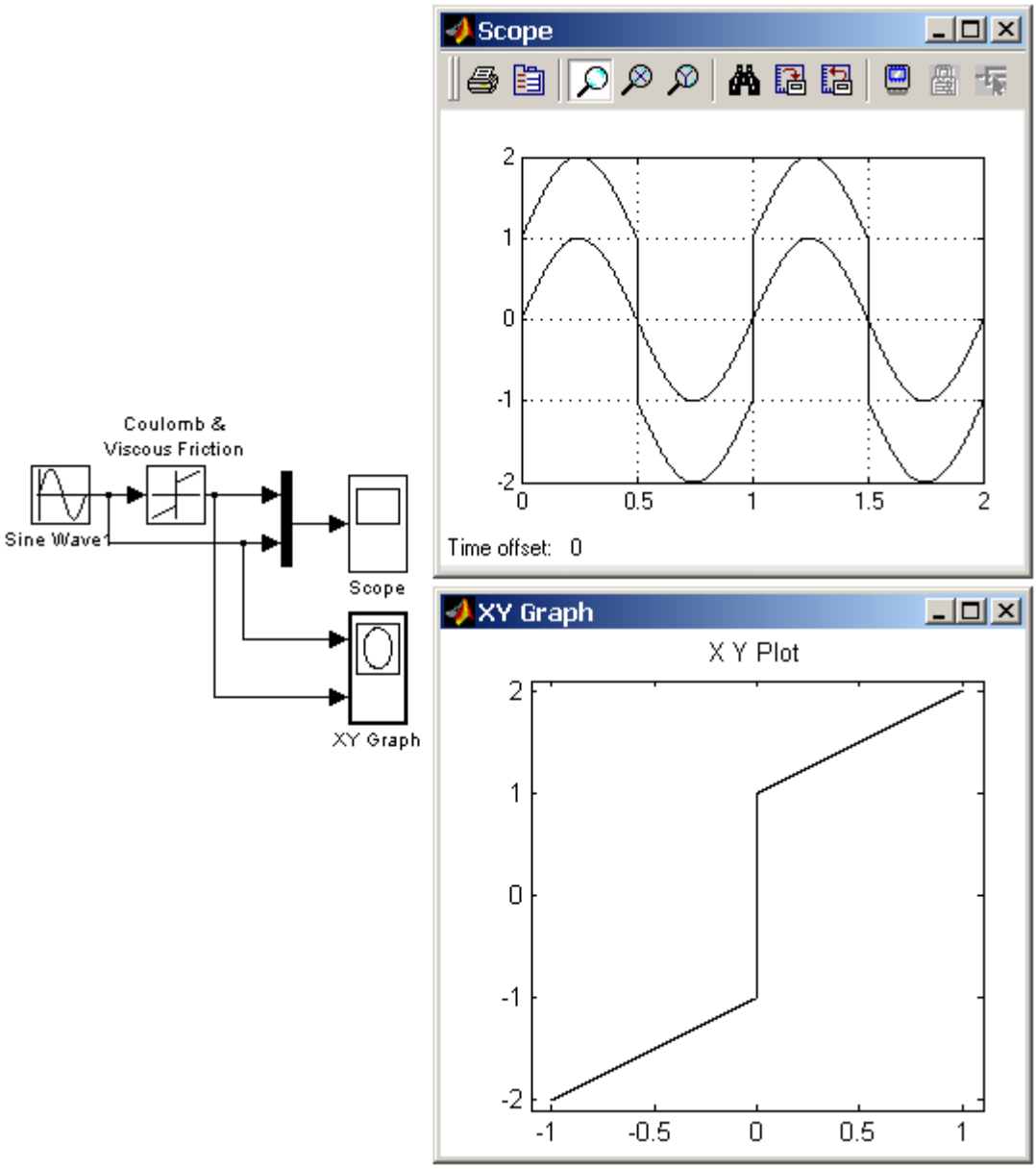


Рис. 3.33.

8. Блок люфта Backlash.

Назначение:

Моделирует нелинейность типа «люфт».

Параметры:

1. **Deaband width** – Ширина люфта.
2. **Initial output** – Начальное значение выходного сигнала.

Сигнал на выходе будет равен заданному значению **Initial output**, пока входной сигнал при возрастании не достигнет значения **(Deaband width)/2** (где **U** – входной сигнал), после чего выходной сигнал будет равен **U–(Deaband width)/2.** После того как произойдет смена направления изменения входного сигнала, он будет оставаться неизменным, пока входной сигнал не изменится на величину **(Deaband width)/2,** после чего выходной сигнал будет равен **U+(Deaband width)/2**.

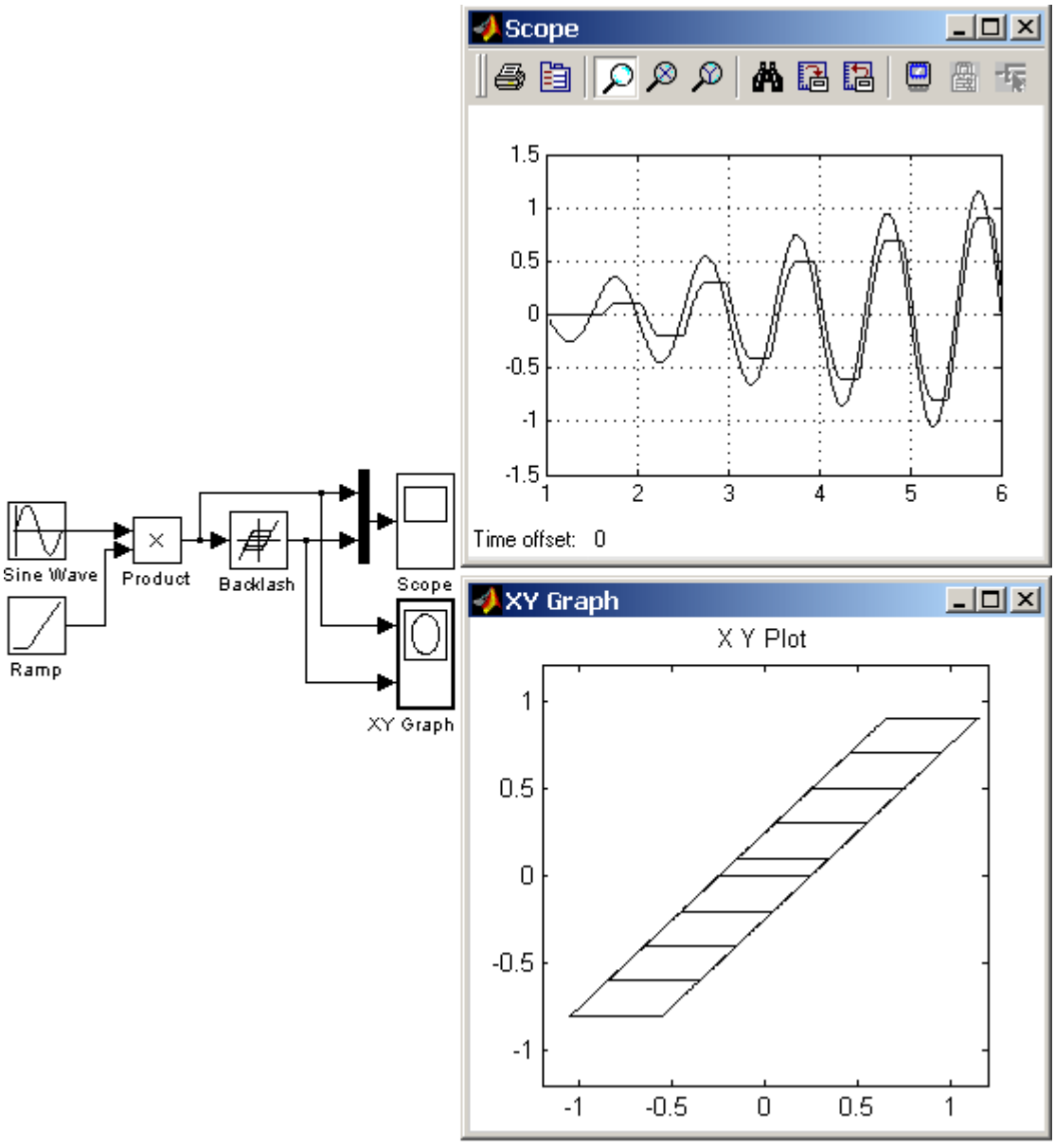


Рис. 3.34.

9. Блок многовходового переключателя Multiport Switch.

Назначение:

Выполняет переключение входных сигналов по сигналу управления, задающему номер активного входного порта.

Параметры:

1. **Number of inputs –** Количество входов.

Блок многовходового переключателя **Multiport Switch** пропускает на выход сигнал с того входного порта, номер которого равен текущему значению управляющего сигнала. Если управляющий сигнал не является сигналом целого типа, то блок **Multiport Switch** производит отбрасывание дробной части числа, при этом в командном окне **Matlab** появляется предупреждающее сообщение.

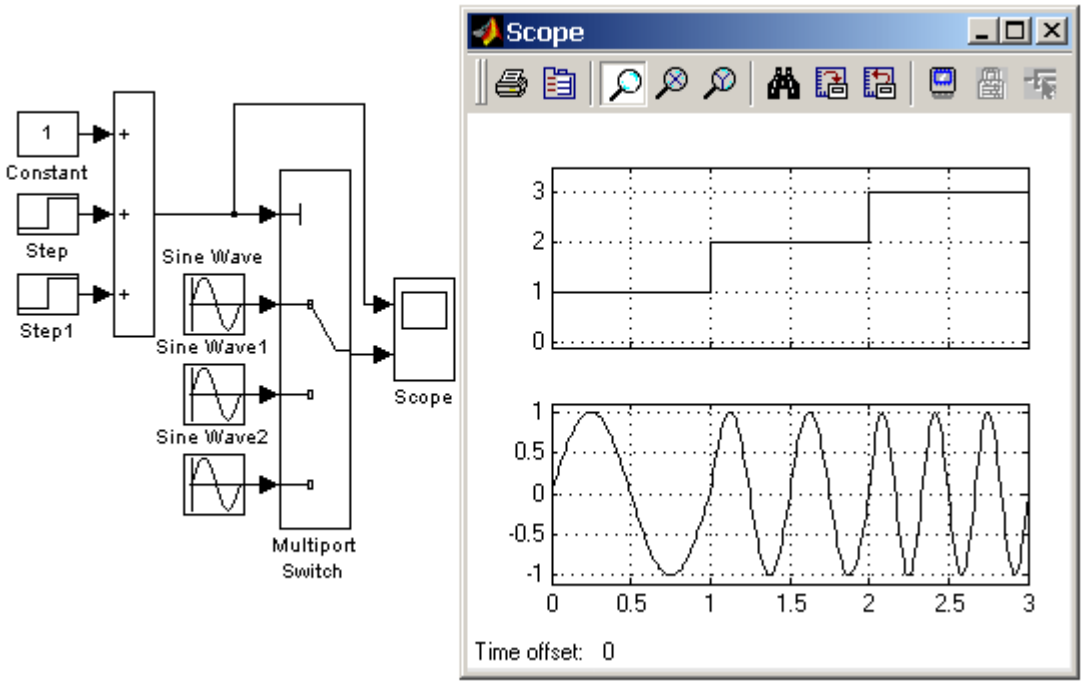


Рис. 3.35.

10. Блок ручного переключателя Manual Switch.

Назначение:

Выполняет переключение входных сигналов по команде пользователя.

Параметры: Нет.

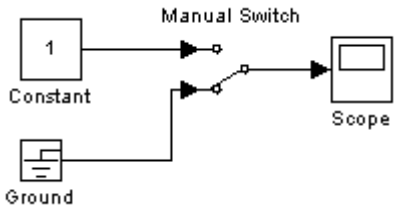


Рис. 3.36.

Командой на переключение является двойной щелчок левой клавишей «мыши» на изображении блока. При этом изображение блока изменяется, показывая, какой входной сигнал в данный момент проходит на выход блока. Переключение блока можно выполнять как до начала моделирования, так и в процессе расчета.

3.9. Math – блоки математических операций

1. Блок вычисления модуля Abs.

Назначение:

Выполняет вычисление абсолютного значения величины сигнала.

2. Блок вычисления суммы Sum.

Назначение:

Выполняет вычисление суммы текущих значений сигналов.

Параметры:

1. **Icon shape** – Форма блока. Выбирается из списка.
   - **round** – окружность;
   - **rectangular** – прямоугольник.
2. **List of sign** – Список знаков.
3. **Saturate on integer overflow** (флажок) – Подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно.

Количество входов и операция (сложение или вычитание) определяется списком знаков параметра **List of sign**, при этом метки входов обозначаются соответствующими знаками. В параметре **List of sign** можно также указать число входов блока. В этом случае все входы будут суммирующими.

Если количество входов блока превышает 3, то удобнее использовать блок **Sum** прямоугольной формы.

Блок может использоваться для суммирования скалярных, векторных или матричных сигналов. Типы суммируемых сигналов должны совпадать.

Если в качестве списка знаков указать цифру 1 (один вход), то блок можно использовать для определения суммы элементов вектора.

3. Блок определения знака сигнала Sign.

Назначение:

Определяет знак входного сигнала.

Параметры:

Нет.

Блок работает в соответствии со следующим алгоритмом:

- Если входной сигнал блока положителен, то выходной сигнал равен 1.
- Если входной сигнал блока отрицателен, то выходной сигнал равен -1.
- Если входной сигнал блока равен 0, то выходной сигнал так же равен 0.

4. Блок умножения Product.

Назначение:

Выполняет вычисление произведения текущих значений сигналов.

Параметры:

1. **Number of inputs** – Количество входов. Может задаваться как число или как список знаков. В списке знаков можно использовать знаки * (умножить) и / (разделить).

2. **Multiplication** – Способ выполнения операции. Может принимать значения (из списка):
   - **Element-wise** – Поэлементный.
   - **Matrix** – Матричный.
3. **Saturate on integer overflow** (флажок) – Подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно.

Если параметр **Number of inputs** задан списком, включающим кроме знаков умножения также знаки деления, то метки входов будут обозначены символами соответствующих операций.

Блок может использоваться для операций умножения или деления скалярных векторных или матричных сигналов. Типы входных сигналов блока должны совпадать. Если в качестве количества входов указать цифру 1 (один вход), то блок можно использовать для определения произведения элементов вектора.

5. Ползунковый регулятор Slider Gain.

Назначение:

Обеспечивает изменение коэффициента усиления в процессе расчета.

Параметры:

1. **Low** – Нижний предел коэффициента усиления.
2. **High** – Верхний предел коэффициента усиления.

Для изменения коэффициента усиления блока **Slider Gain** необходимо передвинуть ползунок регулятора. Перемещение ползунка вправо приведет к увеличению коэффициента усиления, перемещение влево – к уменьшению. Изменение коэффициента усиления будет выполняться в пределах диапазона, заданного параметрами **Low** и **High**.

Если щелкнуть с помощью «мыши» на левой или правой стрелках шкалы регулятора, то коэффициент усиления изменится на 1% от установленного диапазона. Если щелкнуть с помощью «мыши» на самой шкале регулятора слева или справа от ползунка, то коэффициент усиления изменится на 10% от установленного диапазона. Можно также просто задать требуемое значение коэффициента в среднем окне блока.

Блок может выполнять поэлементное усиление векторного или матричного сигнала. Входной сигнал может быть комплексным.

6. Усилители Gain и Matrix Gain.

Назначение:

Выполняют умножение входного сигнала на постоянный коэффициент.

Параметры:

1. **Gain** – Коэффициент усиления.
2. **Multiplication** – Способ выполнения операции. Может принимать значения (из списка):

- **Element-wise K·u**– Поэлементный.
- **Matrix K·u** – Матричный. Коэффициент усиления является левосторонним операндом.
- **Matrix u·K** – Матричный. Коэффициент усиления является правосторонним операндом.

3. **Saturate on integer overflow** (флажок) – Подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно.

Блоки усилителей **Gain** и **Matrix Gain** есть один и тот же блок, но с разными начальными установками параметра **Multiplication**.

Параметр блока **Gain** может быть положительным или отрицательным числом, как больше, так и меньше 1. Коэффициент усиления можно задавать в виде скаляра, матрицы или вектора, а также в виде вычисляемого выражения.

В том случае если парметр **Multiplication** задан как **Element-wise K·u**, то блок выполняет операцию умножения на заданный коэффициент скалярного сигнала или каждого элемента векторного сигнала. В противном случае блок выполняет операцию матричного умножения сигнала на коэффициент, заданный матрицей.

По умолчанию коэффициент усиления является действительным числом типа double.

Для операции поэлементного усиления входной сигнал может быть скалярным, векторным или матричным любого типа, за исключением логического (boolean). Элементы вектора должны иметь одинаковый тип сигнала. Выходной сигнал блока будет иметь тот же самый тип, что и входной сигнал. Параметр блока **Gain** может быть скаляром, вектором или матрицей либого типа, за исключением логического (boolean).

При вычислении выходного сигнала блок Gain использует следующие правила:

- Если входной сигнал действительного типа, а коэффициент усиления комплексный, то выходной сигнал будет комплексным.
- Если тип входного сигнала отличается от типа коэффициента усиления, то **Simulink** пытается выполнить приведение типа коэффициента усиления к типу входного сигнала. В том случае, если такое приведение невозможно, то расчет будет остановлен с выводом сообщения об ошибке. Такая ситуация может возникнуть, например, если входной сигнал есть беззнаковое целое (uint8), а параметр **Gain** задан отрицательным числом.

7. Блок скалярного умножения Dot Product.

Назначение:

Выполняет вычисление скалярного произведения (свертку) двух векторов.

Параметры: Нет.

8. Блок вычисления математических функций Math Function.

Назначение:

Выполняет вычисление математической функции.

Параметры:

1. **Function** – Вид вычисляемой функции (выбирается из списка):
   - **exp** – Экспоненциальная функция;
   - **log** – Функция натурального логарифма;
   - **10^u** – Вычисление степени 10;
   - **log10** – Функции логарифма;
   - **magnitude^2** – Вычисление квадрата модуля входного сигнала;
   - **square** – Вычисление квадрата входного сигнала;
   - **sqrt** – Квадратный корень;
   - **pow** – Возведение в степень;
   - **conj** – Вычисление комплексно-сопряженного числа;
   - **reciprocal** – Вычисление частного от деления входного сигнала на 1;
   - **hypot** – Вычисление корня квадратного из суммы квадратов входных сигналов (гипотенузы прямоугольного треугольника по значениям катетов);
   - **rem** – Функция, вычисляющая остаток от деления первого входного сигнала на второй;
   - **mod** – Функция, вычисляющая остаток от деления с учетом знака;
   - **transpose** – Транспонирование матрицы;
   - **hermitian** – Вычисление эрмитовой матрицы.
2. **Output signal type** – Тип выходного сигнала (выбирается из списка):
   - **auto** – Автоматическое определение типа;
   - **real** –Действительный сигнал;
   - **complex** – Комплексный сигнал.

9. Блок вычисления тригонометрических функций Trigonometric Function.

Назначение:

Выполняет вычисление тригонометрической функции.

Параметры:

1. **Function** – Вид вычисляемой функции (выбирается из списка):
   sin, cos, tan, asin, acos, atan, atan2, sinh, cosh и **tanh**.
2. **Output signal type** – Тип выходного сигнала (выбирается из списка):
   - **auto** – Автоматическое определение типа.
   - **real** –Действительный сигнал.
   - **complex** – Комплексный сигнал.

10. Блок вычисления действительной и (или) мнимой части комплексного числа Complex to Real-Imag.

Назначение:

Вычисляет действительную и (или) мнимую часть комплексного числа.

Параметры:

1. **Output** – Выходной сигнал (выбирается из списка):
   - **Real** – Действительная часть.
   - **Image** – Мнимая часть.
   - **RealAndImage** – Действительная и мнимая часть.

11. Блок вычисления модуля и (или) аргумена комплексного числа Complex to Magnitude-Angle.

Назначение:

Вычисляет модуль и (или) аргумент комплексного числа.

Параметры:

1. **Output** – Выходной сигнал (выбирается из списка):
   - **Magnitude** – Модуль.
   - **Angle** – Аргумент.
   - **MagnitudeAndAngle** – Модуль и аргумент.

12. Блок вычисления комплексного числа по его действительной и мнимой части Real-Imag to Complex.

Назначение:

Вычисляет комплексное число по его действительной и мнимой части.

Параметры:

1. **Input** – Входной сигнал (выбирается из списка):
   - **Real** – Действительная часть.
   - **Image** – Мнимая часть.
   - **RealAndImage** – Действительная и мнимая часть.
2. **Image part** – Мнимая часть. Параметр доступен, если параметр Input объявлен как Real.
3. **Real part** – Действительная часть. Параметр доступен, если параметр **Input** объявлен как **Image**.

13. Блок вычисления комплексного числа по его модулю и аргументу Magnitude-Angle to Complex.

Назначение:

Вычисляет комплексное число по его модулю и аргументу.

Параметры:

1. **Input** – Входной сигнал (выбирается из списка):
   - **Magnitude** – Модуль.
   - **Angle** – Аргумент.
   - **MagnitudeAndAngle** – Модуль и аргумент.
2. **Angle** – Аргумент. Параметр доступен, если параметр **Input** объявлен как **Magnitude**.

3. **Magnitude** – Модуль. Параметр доступен, если параметр **Input** объявлен как **Angle**.

14. Блок определения минимального или максимального значения MinMax.

Назначение:

Определяет максимальное или минимальное значение из всех сигналов, поступающих на его входы.

Параметры:

1. **Function** – Выходной параметр. Выбирается из списка:
   - **min** – Минимальное значение.
   - **max** – Максимальное значение.
2. **Number of input ports** – Количество входных портов.

Входные сигналы блока могут быть скалярными или векторными. Блок определяет максимальное или минимальное значение из всех скалярных сигналов, поступающих на его входы. Если входные сигналы являются векторными, то блок выполняет поэлементную операцию поиска минимального или максимального значения. В этом случае размерности векторов должны совпадать. Если количество входных портов блока задано равным 1, то блок может использоваться для нахождения минимального или максимального значения во входном векторе.

15. Блок округления числового значения Rounding Function.

Назначение:

Выполняет операцию округления числового значения.

Параметры:

1. **Function** – Способ округления (выбирается из списка):
   - **floor** – Округление до ближайшего меньшего целого.
   - **ceil** – Округление до ближайшего большего целого.
   - **round** – Округление до ближайшего целого.
   - **fix** – Округление отбрасыванием дробной части.

16. Блок вычисления операции отношения Relational Operator.

Назначение:

Блок сравнивает текущие значения входных сигналов.

Параметры:

1. **Relational Operator** – Тип операции отношения (выбирается из списка):

   == – Тождественно равно.

   ~= – Не равно.

   < – Меньше.

   <= – Меньше или равно.

   >= – Больше или равно.

   > – Больше.

В операции отношения первым операндом является сигнал, подаваемый на первый (верхний) вход блока, а вторым операндом – сигнал, подаваемый на

второй (нижний) вход. Выходным сигналом блока является 1, если результат вычисления операции отношения есть «ИСТИНА» и 0, если результат – «ЛОЖЬ».

17. Блок логических операций Logical Operation.

Назначение:

Реализует одну из базовых логических операций.

Параметры:

1. **Operator** – Вид логической операции (выбирается из списка):
   - **AND** – Логическое умножение (операция И).
   - **OR** – Логическое сложение (операция ИЛИ).
   - **NAND** – Операция И-НЕ.
   - **NOR** – Операция ИЛИ-НЕ.
   - **XOR** – Исключающее ИЛИ (операция сложения по модулю 2).
   - **NOT** – Логическое отрицание (НЕ).
2. **Number of input ports** – Количество входных портов.

Выходным сигналом блока является 1, если результат вычисления логической операции есть «ИСТИНА» и 0, если результат – «ЛОЖЬ».

3.10. Signal&Systems – блоки преобразования сигналов и вспомогательные блоки

1. Мультиплексор Mux.

Назначение:

Объединяет входные сигналы в вектор.

Параметры:

1. **Number of Inputs** – Количество входов.
2. **Display option** – Способ отображения. Выбирается из списка:
   - **bar** – Вертикальный узкий прямоугольник черного цвета.
   - **signals** – Прямоугольник с белым фоном и отображением меток входных сигналов.
   - **none** – Прямоугольник с белым фоном без отображения меток входных сигналов.

Входные сигналы блока могут быть скалярными и (или) векторными.

2. Демультиплексор Demux.

Назначение:

Разделяет входной векторный сигнал на отдельные составляющие.

Параметры:

1. **Number of Outputs** – Количество выходов.
2. **Bus Selection Mode** (флажок) – Режим разделения векторных сигналов.

Входным сигналом в обычном режиме является вектор, сформированный любым способом. Выходными сигналами являются скаляры или векторы, количество которых и размерность определяется параметром Number of Outputs и размерностью входного вектора.

3. Блок шинного формирователя Bus Creator.

Назначение:

Формирует шину из сигналов различных типов.

Параметры:

1. **Signal naming options** – Способ именования сигнала. Выбирается из списка:
   - **Inherit bus signal names from input ports** – Наследовать имена входных сигналов.
   - **Require input signal names to match signals below** – Требуется ввести имена сигналов.
2. **Number of inputs ports** – Количество входных портов.
3. **Signals in bus** – Список сигналов, объединяемых в шину.
4. **Rename selected signal** – Новое имя выделенного сигнала. Параметр доступен, если выбрана опция **Require input signal names to match signals below**.

Блок позволяет объединять любые сигналы (векторные, матричные, комплексные, действительные и целые разных типов) в единую шину. Такая шина позволяет сократить количество соединительных линий в модели. Для разделения шины на отдельные составляющие необходимо использовать блок **Bus Selector**.

Окно параметров блока позволяет отыскать блок, который является источником сигнала. Для такого поиска необходимо выделить название сигнала в списке **Signals in bus** и нажать с помощью мыши кнопку **Find**. Блок, являющийся источником выбранного сигнала, будет выделен цветом.

4. Блок шинного селектора Bus Selector.

Назначение:

Выделяет из шины требуемые сигналы.

Параметры:

1. **Signals in the bus** – Имеющиеся в шине сигналы (входные сигналы).
2. **Selected signals** – Выделенные сигналы (выходные сигналы).
3. **Muxed output** (флажок) – Объединение выходных сигналов в один.

Шина может быть сформирована блоком **Mux** или **Bus Creator**.

Для извлечения сигнала из шины необходимо открыть окно параметров блока, выделить сигнал в окне **Signals in the bus** и, с помощью кнопки **Select**, скопировать имя сигнала в окно **Selected signals**. Для удаления сигнала из списка **Selected signals** необходимо выделить его имя в правом списке окна параметров блока и, затем, воспользоваться кнопкой **Remove**.

5. Блок определения момента пересечения порогового значения Hit Crossing.

Назначение:

Определяет момент времени, когда входной сигнал пересекает заданное пороговое значение.

Параметры:

1. **Hit crossing offset** – Порог. Значение, пересечение которого входным сигналом требуется идентифицировать.
2. **Hit crossing direction** – Направление пересечения. Выбирается из списка:
   - **rising** – Возрастание.
   - **failing** – Убывание.
   - **either** – Оба направления.
3. **Show output port** (флажок) – Показать выходной порт. В том случае, если этот флажок снят, то точка пересечения сигналом порогового уровня находится, но выходной сигнал блоком не генерируется.

В момент пересечения порогового уровня блок вырабатывает единичный сигнал длительностью в один шаг модельного времени.

4. Моделирование систем управления (Model vision studium)

Model Vision Studium (MVS) – это интегрированная графическая оболочка для быстрого создания интерактивных визуальных моделей сложных динамических систем и проведения вычислительных экспериментов с ними. Пакет разработан исследовательской группой «Экспериментальные объектные технологии» в Санкт-Петербургском Государственном Техническом Университете.

Пакет предназначен для численного моделирования гибридных систем. Гибридная система – это специальная математическая модель реальных объектов, обладающих одновременно «непрерывными» и «дискретными» свойствами. Такие системы достаточно трудно формализовать, так как необходим единый язык для описания как непрерывных, так и дискретных аспектов поведения, но еще труднее корректно численно решить возникающую при этом математическую задачу [8].

Проектировщики MVS исходили из того, что современный язык моделирования должен быть в максимальной степени визуальным и поддерживать технологию объектно-ориентированного моделирования. Корректное численное решение должно получаться автоматически. Таким образом, анализ свойств решаемой математической задачи и выбор и настройка метода решения должны выполняться именно пакетом, а не пользователем. Пользователь должен иметь возможность активно вмешиваться в ход вычислительного эксперимента, и, при необходимости, получать о решении как можно больше дополнительной информации.

Работа пакета MVS основана на использовании нового типа объекта – активного динамического объекта и специальной формы наглядного представления гибридного поведения – карты поведения. Разработчики стремились создать простой и наглядный инструмент для моделирования гибридных систем, ориентированный на широкий круг прикладных пользователей [9].

4.1. Технология моделирования в MVS

Как и в большинстве других современных графических инструментов моделирования, в основе технологии MVS лежит понятие виртуального лабораторного стенда. На этом стенде размещаются различные виртуальные блоки моделируемой системы – вновь создаваемые и стандартные, такие как «генераторы сигналов», «измерительные приборы», «устройства отображения», соединенные виртуальными «кабелями». Вся виртуальная аппаратура функционирует независимо и параллельно, подобно ее физическим двойникам «в металле».

Для получения виртуального стенда необходимо описать моделируемую систему на входном языке пакета и создать соответствующий этому описанию программный код, выполнение которого компьютером и будет восприниматься как работа стенда.

Интегрированная оболочка пакета представляет собой многооконную среду, позволяющую редактировать проект, автоматически преобразовывать графическое описание модели в текстовое и текстовое в графическое, подключать библиотеки классов, создавать свои библиотеки классов, создавать выполняемые модели и запускать их, а также запускать специальные подсистемы (оптимизации и символического анализа) [9].

Предполагается, что каждой модели (проекту) соответствует определенная папка, в которой хранятся файл внутреннего представления проекта (mvb), файлы установок проекта и выполняемой модели (ini), а также картинки для анимации, DLL пользователя и т. п. Описание проекта и библиотек классов хранится в виде дерева объектов в объектно-ориентированной базе данных MVBase (отдельный файл с расширением mvb на каждый проект и библиотеку классов). Библиотеки классов (за исключением стандартной библиотеки SysLib) являются обычными проектами, их могут создавать и редактировать пользователи.

Описание проекта пользователь может вводить и редактировать как в визуальном, так и в текстовом виде. При открытии в интегрированной среде какого-либо проекта его внутреннее представление автоматически разворачивается в визуальное представление. В любой момент с помощью специальной команды может быть получено текстовое описание проекта на специальном языке Model Vision Language (MVL), включающее в себя два текстовых файла: собственно функциональное описание (расширение mvl) и описание визуальных элементов (расширение pra). Импорт проекта из текстового представления осуществляется специальным MVL-компилятором.

Описание проекта включает в себя описание классов устройств, глобальных констант и алгоритмических процедур и функций, а также описание конкретной конфигурации виртуального стенда, с которой будет проводиться вычислительный эксперимент. Предполагается, что виртуальный стенд является устройством-контейнером TestBench – экземпляром предопределенного класса _cTestBench. Пользователю необходимо поместить в его локальную структуру конкретные локальные устройства – экземпляры классов, определенных в данном проекте или импортируемых из подключенных к проекту библиотек классов. Стандартная библиотека классов SysLib, включающая определения типовых блоков (линейные и нелинейные блоки, генераторы сигналов и т. д.), подключена к любому проекту по умолчанию. При создании выполняемой модели программный код создается только для классов, реально используемых (прямо или косвенно, через другие классы) в TestBench.

Все визуальные редакторы по завершении ввода какой-либо законченной конструкции немедленно проверяют ее синтаксическую и семантическую правильность в контексте уже существующего описания и при обнаружении ошибок выводят соответствующие сообщения.

При генерации выполняемой модели сначала проводится полный комплексный контроль классов, используемых в TestBench, а затем для каждого класса генерируется соответствующий программный модуль на

промежуточном языке программирования, и в зависимости от типа модели генерируется соответствующий главный модуль программы. Затем полученная программа компилируется с помощью компилятора командной строки для промежуточного языка. На этом этапе к сгенерированным модулям присоединяются стандартные модули промежуточного языка и библиотека периода исполнения (RTL) пакета MVS для данного промежуточного языка.

В MVS 3.0 возможны три типа выполняемых моделей:

- визуальная интерактивная модель в виде 32-разрядного приложения для MS Windows;
- «скрытая» модель в виде 32-разрядной DLL для MS Windows;
- визуальная интерактивная модель в виде Java-приложения, выполняемая на любой платформе, где имеется виртуальная машина Java. В версии 3.0 визуальная Java-модель не поддерживает векторных и матричных переменных, а также анимации.

При генерации моделей для Windows в качестве промежуточного языка используется Object Pascal 10.0 (Borland Delphi 3). Необходимые для компиляции модули устанавливаются автоматически при инсталляции пакета MVS.

При генерации Java-модели пользователь должен предварительно установить на своем компьютере JDK 1.2 или более позднюю версию (этот инструментальный пакет бесплатный, его можно найти, например, на сайте фирмы JavaSoft).

В описании проекта пользователь может употреблять собственные внешние процедуры и функции, программную реализацию которых он должен выполнить в соответствующих DLL или Java-классах.

В пакете MVS предусмотрена возможность установки специальных подсистем (ToolBoxes), не входящих в стандартный комплект. Такими в настоящий момент являются подсистема оптимизации и подсистема символического анализа. При обращении к специальной подсистеме интегрированная среда сначала автоматически генерирует «скрытую» модель в виде DLL, а затем запускает соответствующую программу.

4.2. Блоки и связи

Основным «строительным элементом» описания в MVS 3.0 является блок, называемый устройством. Устройство – это некоторый активный объект, функционирующий параллельно и независимо от других объектов в непрерывном времени. В общем случае в описании устройства содержатся следующие элементы: входы, выходы, параметры конструктора, переменные состояния, поведение, локальная структура. Входы, выходы и переменные состояния являются фазовыми переменными, и все вместе составляют фазовый вектор устройства.

Типы данных включают в себя скалярные и регулярные. Скалярными могут быть вещественные, целые, булевский, перечислимые, символьный и строковый типы. Регулярными являются матрицы и векторы с вещественными

элементами. Для передачи информации о дискретных событиях используется специальный тип – сигнал.

Устройства могут соединяться между собой однонаправленными функциональными связями и входить в состав других устройств, образуя иерархическую структуру связей. Следует отметить, что однонаправленность связей является осознанным ограничением, вызванным проблемами с получением численного решения для композиции гибридных блоков.

4.3. Поведение

Предполагается, что поведение любого блока является гибридным. Для задания гибридного поведения авторами пакета выбран формализм гибридного автомата, как самый наглядный и мощный (отрицательным следствием этого выбора являются некоторые проблемы с численным решением, но мы предполагаем, что эти проблемы в перспективе разрешимы).

Гибридным автоматом называется граф переходов, узлам которого приписаны некоторые непрерывные отображения, а дугам – условия переходов и выполняемые действия. В настоящее время для формального описания дискретных "машин состояний" стандартом de facto стала «карта состояний» (statechart), придуманная Д. Харелом и используемая в стандарте UML. Карта состояний, узлам которой приписаны некоторые непрерывные отображения, называется гибридной картой состояний. Она представляет собой простую и очень наглядную форму визуального представления смены поведений.

В MVS используется специальное ограничение гибридной карты состояний, называемое картой поведения.

Карта поведения (bihavior chart или B-chart) – это ориентированный граф, в котором узлам приписываются некоторые локальные поведения, а дугам, называемым переходами, – условия перехода от одного поведения к другому и выполняемые при этом действия. Узел, в котором система находится в каждый конкретный момент времени, называется текущим. Один из узлов должен быть предварительно помечен как начальный, он автоматически становится текущим при создании карты состояний. Соответствующее ему начальное поведение создается при создании экземпляра устройства. Смена текущего узла происходит в результате срабатывания переходов. Когда узел становится текущим, создается экземпляр приписанного ему локального поведения. Созданный экземпляр уничтожается, как только узел перестает быть текущим. Именно в этом и состоит отличие карты поведения от карты состояний – в последней локальное поведение существует всегда и можно вернуться в его текущее состояние (history indicator).

Локальное поведение может быть описано как: непрерывное поведение; карта поведения (в этом случае узел называется гиперузлом); пустое поведение (NULL).

Графический образ карты поведения позволяет в наглядной форме представлять множества допустимых локальных поведений устройства, их области определения в фазовом пространстве и времена переходов от одного локального поведения к другому. При описании локальных поведений

предусмотрена возможность вводить локальные переменные (аналогичные локальным переменным программных единиц).

В общем случае переход T из начального узла Nb в конечный узел Ne характеризуется: охраняющим предикатом G, запускающим событием E, и действиями A. Возможны три типа запускающего события:

- Некоторое логическое условие стало истинным (change event). Поступил внешний сигнал (signal event).
- Истекло определенное время после того как начальный узел Nb стал текущим (time event). Семантика перехода следующая. Если узел Nb является текущим и предикат G истинен или отсутствует, переход T становится открытым, в противном случае переход закрыт. Если событие E не указано, то открытый переход немедленно срабатывает. Если указано событие E, то открытый переход сработает только при его появлении и истинности предиката G (до появления события E переход может закрыться, если предикат G перестает быть истинным или узел Nb текущим). Срабатывание представляет собой приведенную ниже последовательность мгновенных действий:

выполняется мгновенная последовательность выходных действий узла Nb;

- узел Nb перестает быть текущим;
- выполняется последовательность действий A;
- узел Ne становится текущим;

выполняется мгновенная последовательность входных действий узла Ne.

В действиях переходов и узлов возможно использование алгоритмических функций и процедур, а в правых частях уравнений и формул – использование алгоритмических функций, задаваемых либо с помощью встроенного алгоритмического языка, подмножества языка Ada, либо во внешних программных модулях.

В частном случае устройства с чисто дискретным поведением всем узлам карты поведения следует приписать пустые локальные поведения и тогда она превращается в обычную карту состояний (с указанным выше отличием). Элемент с чисто непрерывным поведением трактуется как гибридный с картой поведения, состоящей из единственного узла, которому приписано непрерывное поведение. Таким образом, дискретные аспекты поведения в MVS отражаются с помощью хорошо знакомого языка карт состояния, а непрерывные аспекты – с помощью привычного языка систем уравнений и формул.

4.4. Классы и экземпляры

В MVS существует два вида классов – классы устройств (блоков) и локальные классы поведений, имеющие смысл только в контексте соответствующего класса блока.

Статический экземпляр локального устройства создается автоматически при создании экземпляра составного устройства. Статический экземпляр главного устройства модели (TestBench) создается исполняющей системой при прогоне модели и уничтожается по его окончании. Динамические экземпляры устройств в версии 3.0 не поддерживаются.

Динамический экземпляр локального поведения в узле карты поведения создается автоматически, когда этот узел становится текущим, и уничтожается, когда этот узел перестает быть текущим.

Новый класс устройств может наследовать свойства другого класса. Все устройства являются потомками предопределенного класса coevice, которому приписаны все предопределенные соглашения о взаимодействии с исполняющей системой MVS.

В классе-потомке (субклассе, производном классе) вы можете добавлять новые элементы описания (новые параметры, фазовые переменные, константы, алгоритмические процедуры и функции, локальные поведения, локальные устройства и связи), но не можете удалить никакой элемент, определенный в классе-предке (суперклассе, базовом классе). Вы можете также добавить новые узлы и переходы в унаследованной карте поведения (в этом случае автоматически появляется поведение-потомок). В карте поведения вы можете переопределить локальное поведение в узле, входные и выходные действия в узле, условия и действия в переходе.

В рамках версии 3.0 также планируется доработка, связанная с трактовкой анимации как элемента описания класса устройств, а не экземпляра (см. разд. «Визуальная модель»).

4.5. Выполняемая модель

Как отмечалось, в MVS существуют выполняемые модели двух видов:

- визуальная модель;
- «скрытая» модель (в данный момент реализована только для Windows).

Выполняемая модель любого вида является независимым программным модулем и может использоваться как самостоятельная программа.

4.6. Визуальная модель

Образом визуальной интерактивной модели является испытательный стенд, на котором со всех «приборов» сняты крышки, так что пользователь может наблюдать процесс их функционирования и в любой момент изменять настройки. В визуальной модели пользователь может видеть все и во все вмешиваться (естественно, визуальная модель требует значительно больших вычислительных ресурсов, чем «скрытая»).

Визуальная модель для Windows – это 32-разрядное приложение (exe), а для Java – Java-приложение.

Вычислительный эксперимент с визуальной моделью представляет совокупность прогонов модели от начального состояния (модельное время равно 0, переменные имеют исходные начальные значения) до конечного. Прогон модели включает в себя точки останова, в которых модельное время не изменяется, и интервалы выполнения, в течение которых имитируется функционирование модели в модельном времени. Переход из точки останова к выполнению, от выполнения к точке останова, а также возврат к начальному состоянию модели производятся с помощью соответствующих команд.

Вычисление значений фазовых переменных модели производится в некотором модельном времени. Текущее значение модельного времени в условных единицах измерения отображается в левом углу инструментальной панели. В то же время пользователь наблюдает за ходом прогона модели и интерактивно вмешивается в него в реальном физическом времени. Соотношение модельного и физического времени зависит от производительности процессора и сложности совокупного поведения моделируемой системы и в общем случае изменяется по ходу прогона. Отображение значений модельного времени и значений переменных в окнах фазового вектора и матричных переменных, а также ввод значений от интерактивных анимационных компонентов производится в реальном времени с частотой 2–10 Гц. Начальное значение частоты вывода 10 Гц. Если компьютер медленный или уравнений очень много, частота вывода понижается, но остается не менее 2 Гц.

Моделирование может осуществляться «как получается» или с выдерживанием определенного соотношения модельного и реального времени. По умолчанию используется первый режим. В этом случае прогон модели идет так быстро, как позволяет мощность процессора. Переменное соотношение скорости изменения модельного и реального времени неудобно при наблюдении анимации или в случае работы с реальной аппаратурой. В этом случае следует установить второй режим и задать определенное соотношение скоростей модельного и реального времени. В этом режиме значение модельного времени отображается красным цветом, если на данном компьютере не удается выдержать заданное соотношение. В любом случае происходит периодическая синхронизация процесса моделирования с течением реального времени.

Для задания внешних воздействий в ходе прогона пользователь может создать план прогона – линейную последовательность в модельном времени операторов присваивания значений переменным модели, посылки сигналов и вывода сообщений.

Отладочные возможности визуальной модели включают в себя задание точки останова по истинности некоторого логического выражения, по срабатыванию перехода (заданного или любого) и по входу в заданный узел карты поведения.

При появлении ошибки в ходе вычислений определяется блок и локальное поведение, где она произошла, а в отладочном режиме (установлен по

умолчанию) дополнительно выводится текстовый вид уравнения или оператора, при выполнении которого она была обнаружена.

В отличие от большинства пакетов блочного моделирования в визуальной модели MVS вспомогательные средства визуализации (диаграммы, аниматоры) не должны входить в описание моделируемой системы, а присоединяются прямо в визуальной модели (можно создать несколько вариантов виртуального стенда и запомнить их в соответствующих установках визуальной модели).

Пользователю доступны следующие стандартные окна: Test Bench (Испытательный стенд); Phase Vector (Фазовый вектор); Behavior (Карта поведения); Time diagram (Временная или фазовая диаграмма); Local structure (Локальная структура); 2D-animation (Двумерная анимация); 3D-animation (Трехмерная анимация); Formula calculator (Калькулятор выражений).

В окне Phase Vector динамически отображаются текущие значения всех фазовых переменных (параметров, входов, выходов и переменных состояния) выбранного устройства. Значение переменной отображается в соответствии с ее типом. Значения векторов и матриц выводятся в виде матричных литералов, которые могут быть представлены либо в строчном виде, либо отображаться с помощью специального окна. Если значение вектора или матрицы не умещается в строке, то это обозначается многоточием в квадратных скобках ([...]). В этом случае значение координат можно посмотреть в специальном окне для отображения «больших» матричных объектов. Постоянные фазовые переменные (переменные, определенные в классе) отображаются черным цветом, временные фазовые переменные (переменные, определенные в текущем локальном поведении), отображаются серым цветом. При помещении курсора мыши на изображение переменной появляется подсказка, соответствующая комментарию к данной переменной, если он был указан в тексте программы. Значения переменных можно изменять по ходу эксперимента. По нажатии клавиши <Enter> или двойному нажатии левой кнопки мыши, а также по специальной команде всплывающего меню происходит переход в диалоговый режим редактирования текущего значения компонента в соответствии с его классом и типом. В качестве нового значения можно задать любое выражение, включающее переменные модели. Для того чтобы присвоить переменной модели новое значение, нужно обеспечить синхронность с процессом моделирования. Точкой синхронизации может оказаться ближайшее дискретное событие, окончание шага интегрирования или точка синхронизации модельного и реального времени

В окне Behavior отображается его динамика: текущий узел, активные переходы и сработавший переход. Текущий узел окрашивается синим цветом. Линии активных переходов ярче пассивных. Линия срабатывающего перехода подсвечивается в момент перехода. Окно главной карты поведения можно открыть с помощью команды всплывающего меню окна Local structure, появляющемся при нажатии правой кнопки мыши на изображении соответствующего устройства. Окно локальной карты поведения, приписанной какому-либо узлу главной карты поведения (изображается двойной линией), можно открыть с помощью двойного щелчка мышью на этом узле или путем

выделения этого узла и нажатия клавиши <Enter> и т. д. по уровням вложенности.

Окно Time diagram представляет собой график, в котором по оси X откладывается либо значение модельного времени (временная диаграмма), либо значение одной из переменных (фазовая диаграмма), а по оси Y– значения переменных. Переменные связываются с окном Time diagram путем их буксировки из окон фазового вектора, матричной переменной или структуры.

Если вы хотите отображать в окне Time diagram переменную векторного или матричного типа, следует дополнительно указать конкретные компоненты. В отличие от многих других пакетов, в нашем случае окно Time diagram строится одновременно с производимыми вычислениями, а не после их завершения. По умолчанию окно является автомасштабируемым, но масштаб можно выбирать и самостоятельно. В окне могут отображаться вещественные переменные, переменные перечислимого типа и сигналы. Все вещественные переменные могут отображаться в едином масштабе или каждая в своем собственном – иногда бывает удобным изображать на одном графике переменные с существенно различными диапазонами значений. Отображение перечислимых переменных не имеет общепринятого стандарта. Мы воспользовались тем, что на наши графики нанесена сетка. Клетки сетки и соответствуют различным значениям переменной перечислимого типа. Для отображения каждого дискретного значения выбираются точки, отстоящие по вертикали от ближайшего значения на одну клетку. Графики, соответствующие переменным перечислимого типа, имеют еще одну особенность. Два и больше таких графиков, вообще говоря, нельзя изображать в одних осях, как мы это делаем для вещественных переменных, т. к. переменные перечислимого типа не сравнимы. Мы, тем не менее, отображаем такие графики в одних осях друг под другом. Сигналы изображаются вертикальными линиями длиной в одну клетку.

При работе с гибридными моделями на временных диаграммах возникают изображения, соответствующие многозначным функциям. То есть одной и той же временной точке могут соответствовать различные значения одной и той же переменной. Это связано с тем, что на графике достаточно сложно отделить точку, соответствующую моменту срабатывания перехода, от точек интервала, где реализуется непрерывное локальное поведение. Например, если переменная сначала изменялась по некоторому закону, а в момент срабатывания перехода T изменяется скачком в действиях перехода, то в точке диаграммы, соответствующей моменту T, будет отображено и последнее значение переменной, соответствующее функции локального поведения, и новое, присвоенное ей в действиях перехода. Окно Time diagram не только показывает графики переменных, но хранит всю информацию, необходимую для их повторного просмотра без проведения повторных вычислений. Все занесенные на диаграмму точки сохраняются, пока окно открыто. Таким образом, из точки приостановки можно вернуться в начало эксперимента или любую другую временную точку, перестроить график в другом масштабе, превратить временную диаграмму в фазовую и наоборот.

В окне Local structure отображается структурная схема выбранного составного устройства. Контуры графических обозначений элементарных устройств обозначаются одинарными линиями, составных – двойными. При прогоне модели изменения значений на линии связи могут отображаться кратковременным выделением цветом (таким образом, непрерывное изменение будет отображаться постоянным выделением линии цветом). При нажатии правой кнопки мыши на изображении устройства появляется всплывающее меню, с помощью команд которого можно открыть окно фазового вектора, локальной структуры или главной карты поведения этого устройства. Окно локальной структуры можно также открыть, дважды щелкнув мышью на его изображении. Таким образом, окно структуры TestBench является своего рода «проводником» по всей модели. При двойном нажатии левой кнопки мыши на изображении входа или выхода появляется диалог изменения значения (моделирование временно приостанавливается). В точке останова при перемещении курсора мыши на изображении входа или выхода выводится окно подсказки с текущим значением этой переменной. С входом или выходом можно связать один из стандартных 2D-анимационных компонентов, среди которых есть интерактивные.

Окно Formula calculator позволяет вычислить в точке останова значения одного или нескольких выражений, в которые могут входить переменные модели.

В визуальной модели пользователь может использовать средства 2D и 3D-анимации.

Визуальная модель одновременно является сервером автоматизации, поддерживающим интерфейс IMVSAuto. Он позволяет осуществлять чтение и запись значений переменных моделей и управление прогоном модели из других приложений.

4.7. «Скрытая» модель

«Скрытая» модель является исключительно вычислительной и не имеет ни окон, ни визуальных элементов управления. Все, что она может, это вычислить значения переменных модели в заданной временной точке, но делает она это значительно быстрее, чем визуальная. В MVS 3.0 «скрытая» модель – это DLL, включающая код модели и часть кода исполняющей системы MVS и экспортирующая набор процедур и функций, обеспечивающих возможность использования этой DLL внешним приложением.

Внешнее приложение имеет возможность получить список всех устройств и переменных модели, прочитать и записать значение переменной, продвинуть модельное время на интервал, продвинуть модельное время до выполнения некоторого условия, начать новый прогон модели, вычислить значение выражения и т. д. Комбинация «скрытой» модели и внешнего приложения очень удобна для вычислительных экспериментов, требующих многократных прогонов модели по определенному алгоритму или при разработке специальной анимации.

Таким образом, MVS разрешает пользователю, предварительно отладив описание моделируемой системы и проверив его адекватность с помощью удобной, но громоздкой и медленной визуальной модели, автоматически создать затем компактную и быструю «скрытую» модель и встроить ее в свое специальное приложение.

4.8. Анимация

Поддержка анимации является одной из важнейших возможностей современных инструментов моделирования. Однако создание сложной анимации даже при использовании стандартной графической библиотеки (такой как OpenGL или DirectX) является сложной задачей даже для специалистов и не по силам подавляющему числу пользователей. Поэтому разработчики пакетов моделирования при определении встроенных средств анимации вынуждены идти на определенные компромиссы.

В визуальных моделях MVS 3.0 поддерживается встроенная 2D- и 3D-анимация (только для Windows).

Имеется набор стандартных 2D-компонентов в стиле LabView (линейные индикаторы, стрелочный индикатор, линейный движок, поворотный регулятор, цветовой индикатор, кнопки), которые можно поместить либо в окно 2D-animation, либо в окно Local structure (структурная схема в этом случае играет роль мнемосхемы).

Окно 2D-animation предназначено для размещения на некоторой фоновой подложке совокупности стандартных анимационных компонентов, снабженных пояснительными надписями. Например, это может быть образ пульта управления или мнемосхемы моделируемой системы с соответствующими индикаторами. Стандартные компоненты размещаются в окне с помощью мыши (аналогично работе в редакторе Delphi). Чтобы связать анимационный компонент с переменными модели, нужно просто выделить нужную переменную в окне фазового вектора или в окне структуры, затем с помощью операции drag-and-drop переместить ее на изображение анимационного компонента. После того текущее значение переменной будет однозначно связано, например, с углом поворота стрелки и при изменении значения переменной в ходе прогона стрелка на индикаторе будет отклоняться. Для интерактивных компонентов, напротив, перемещение с помощью мыши движка или поворот ручки регулятора вызовет (после синхронизации с решателем) изменение значения ассоциированной с регулятором переменной модели и далее – изменение фазовой траектории модели. Таким образом, пользователь может непосредственно с помощью мыши задавать различные внешние воздействия и наблюдать реакцию на них моделируемой системы.

Окно 3D-animation позволяет строить динамические трехмерные изображения, представляющие совокупности трехмерных примитивов (линия, шар, цилиндр, конус и т. д.). Поддержка 3D-анимации требует наличия на компьютере библиотеки OpenGL (в случае Windows NT/98/2000 эта библиотека устанавливается вместе с операционной системой). Изменение параметров сцены, задание требуемого набора трехмерных примитивов и связывание их

параметров с переменными модели осуществляется в окне специального редактора. В любой момент вы можете изменить точку наблюдения, нажав левую кнопку мыши и перемещая ее.

В окне 3D-animation доступны следующие примитивы: линия; тор; стрелка; сфера; спираль; поверхность; цилиндр; брус; текст; конус; четырехугольник.

С помощью этого набора примитивов можно построить достаточно сложные трехмерные конструкции. Для создания более сложной анимации следует использовать внешнее приложение и «скрытую» модель.

4.9. Исполняющая система

Исполняющая система MVS имеет ряд особенностей, связанных, во-первых, с использованием гибридной карты состояний и, во-вторых, с интерактивностью визуальной модели и синхронной визуализацией.

Текущее глобальное непрерывное поведение модели представляет собой суперпозицию локальных непрерывных поведений в текущих узлах всех входящих в нее устройств. Для гибридной модели оно изменяется всякий раз, когда перестает быть текущим, или становится текущим узел с непустым локальным непрерывным поведением. Использование в MVS ориентированных блоков позволяет проводить на стадии компиляции модели локальный анализ каждого непрерывного поведения в отдельности, без учета связей между блоками. Однако глобальный анализ непрерывного поведения всей модели на стадии компиляции потребовал бы анализа всех сочетаний поведений в узлах для всех устройств модели. Поэтому при использовании карты поведения часть работы по анализу неизбежно переносится на этап исполнения. На стадии компиляции выполняется наиболее трудоемкая часть анализа, требующая исходной информации о проекте, и вырабатывается предельно простая для дальнейшего анализа на стадии исполнения промежуточная информация.

Для совокупности формул одного непрерывного поведения, которые не зависят от связей, генерируется одна процедура – метод программного класса, соответствующего поведению. В ее теле находится код локально отсортированной последовательности формул – своего рода макроформула. Для формул, зависящих от связей, генерируются отдельные процедуры. Это позволяет сэкономить время на глобальной сортировке во время исполнения модели, которое возрастает пропорционально как минимум квадрату числа формул. Во время компиляции определяется и передается на этап исполнения перечень входных переменных, являющихся приемниками в связях и входящих в правые части формул. Эта информация необходима для глобальной сортировки формул во время исполнения модели.

На стадии исполнения всякий раз после завершения всех срабатываний переходов в данной временной точке определяется, изменилось ли глобальное непрерывное поведение и, если изменилось, проводится анализ нового поведения (напомним, что на стадии исполнения анализатору доступны только информация о программном коде модели и промежуточная информация, выработанная на стадии компиляции). Целями глобального анализа являются:

- сортировка формул с учетом связей; обнаружение и разрыв алгебраических циклов;
- определение типа глобальной системы уравнений и выбор адекватного численного метода.

Сортировка формул производится на основе промежуточной информации, в которой входные переменные заменяются соответствующими выходными переменными других блоков с учетом реально существующих связей (для динамической структуры может не существовать экземпляров некоторых блоков и, следовательно, всех исходящих от них связей). Макроформулы могут вычисляться предварительно в любом порядке.

Проверка на алгебраические циклы проводится в случае, если два или более локальных поведения, входящих в глобальную суперпозицию, имеют выработанный на стадии компиляции признак наличия «транзитных цепочек». В случае обнаружения алгебраического цикла одна из образующих его связей разрывается и заменяется соответствующим алгебраическим уравнением (для пользователя выводится предупреждающее сообщение), после чего снова проводится проверка на алгебраические циклы до тех пор, пока все алгебраические циклы не будут разорваны. При изменении локальных поведений, входящих в циклы, дополнительные алгебраические уравнения ликвидируются, и связи восстанавливаются.

Исполняющая система определяет вид новой глобальной системы уравнений (алгебраическая, дифференциальная или алгебро-дифференциальная) и выбирает указанный в установках численный метод (см. ниже) для этого вида уравнений. По умолчанию используются «методы-автоматы». В любом случае после дискретного события численный метод заново инициализируется, т. к. в результате мгновенных действий в переходах и узлах может скачком измениться значение правых частей уравнений.

Другой проблемой, связанной с гибридностью модели, является поиск точки переключения непрерывных поведений. Точка переключения может быть обусловлена:

- изменением значения логического предиката дискретного перехода (change event);
- переходом на другую логическую ветвь условного выражения в правых частях уравнений и формул или условного оператора в функциях, вызываемых из правых частей («скрытое» дискретное событие).

При решении текущей системы уравнений на каждом шаге проверяется наличие переключения. Если оно обнаружено, то методом половинного деления находятся решения системы для двух временных точек T0 и T, таких, что в T0 еще нет переключения, а в T оно уже есть, а разность фазовых переменных и времени в этих точках удовлетворяют заданным в установках погрешностям.

В рамках версии 3.0 планируется доработка, связанная с дополнительным анализом характера поведения функций, определяющих предикат change event, для самого распространенного случая, когда предикат является логической комбинацией отношений вещественных выражений. Это позволит находить

точку переключения более быстро и не пропустить переключения при больших значениях шага интегрирования.

Особыми событиями, прерывающими процесс численного решения («решатель» реализован в виде отдельного потока, независимого от визуальной оболочки), являются интерактивное вмешательство или точка визуализации.

При изменении положения интерактивного анимационного элемента или при двойном щелчке мыши на изображении фазовой переменной выставляется специальный флаг, по которому по окончании очередного шага интегрирования процесс решения прерывается, вводится новое значение соответствующей переменной, а затем решение продолжается с новыми начальными условиями.

В отличие от многих пакетов моделирования, в MVS визуализация результатов осуществляется не после, а во время прогона модели. Момент (в модельном времени) вывода очередной точки на временную диаграмму определяется выбранным масштабом по оси времени (его единица измерения пикселов в секунду). Вывод новой точки на фазовой диаграмме, а также обновление окон фазового вектора и анимационных окон и компонентов производится с определенной частотой в реальном времени пользователя (в зависимости от производительности компьютера она регулируется автоматически в пределах 2–10 Гц). Точка визуализации является особым дискретным событием, которое не требует перезапуска численного метода.

5. Моделирование электрических, магнитных и тепловых полей (ELCUT)

ELCUT – это интегрированная диалоговая система программ, позволяющая решать плоские и осесимметричные задачи следующих типов [10]:

- Расчет электрического поля:
    - Электростатическое поле.
    - Электрическое поле постоянных токов.
    - Электрическое поле переменных токов.
- Расчет магнитного поля:
    - Магнитостатическое поле.
    - Магнитное поле переменных токов (с учетом вихревых токов).
    - Магнитное нестационарное поле (с учетом вихревых токов и нелинейных материалов).
- Задачи теплопередачи (расчет температурного поля):
    - Стационарная теплопередача
    - Нестационарная теплопередача (тепловые переходные процессы).
- Задачи механической прочности:
    - Линейный анализ напряженно-деформированного состояния.

5.1. Основные сведения об организации ELCUT

Используя ELCUT, вы работаете с разными типами документов: задачи, геометрические модели, библиотеки свойств материалов и др. Каждый документ открывается в своем отдельном окне внутри главного окна ELCUT. Вы можете одновременно открыть любое окно. Вы можете изменять содержание активного документа, используя меню, расположенного вверху главного окна ELCUT. Содержание меню различно для документов разных типов.

ELCUT использует следующие типы документов:

1. Описание задачи. Соответствует каждой физической задаче, решаемой при помощи ELCUT. Этот документ содержит такие общие характеристики, как тип задачи («Электростатика», «Магнитостатика», «Теплопередача» и пр.), класс модели (плоская или осесимметричная) и т. п., а также имена других документов, связанных с данной задачей.

2. Геометрическая модель. Содержит полное описание геометрии задачи, метки различных ее частей и расчетную сетку конечных элементов. Разные задачи могут использовать общую модель.

3. Физические свойства. Различаются для разных типов задач (Свойства для электростатики, свойства для вихревых токов и т. д.). Эти документы содержат значения свойств материалов, источников поля и граничных условий для разных помеченных геометрических объектов модели. Документ свойств может быть использован как библиотека материалов для различных задач.

Чтобы решить задачу, нужно ассоциировать с ней имена двух документов: модели и физических свойств. Для большего удобства задача может ссылаться

на два документа свойств одновременно: один из них, называемый справочник свойств, содержит свойства часто используемых материалов (библиотека материалов), а другой документ содержит данные, специфичные для данной задачи или группы задач.

Типичная последовательность шагов при решении новой задачи представлена на блок-схеме:

5.2. Обзор основных типов задач

Магнитостатика

Расчет магнитного поля применяется при проектировании и исследовании различных устройств, таких как соленоиды, электрические машины, магнитные экраны, постоянные магниты, реакторы, и т. п. Обычно при расчетах магнитного поля представляют интерес такие величины, как магнитная индукция, напряженность магнитного поля, магнитные силы и моменты, индуктивность, а также потокосцепления с различными обмотками.

Пакет ELCUT может применяться для решения линейных и нелинейных задач магнитостатики в плоской и осесимметричной постановке. Используется формулировка задачи относительно векторного магнитного потенциала. При постановке задачи Вы можете использовать следующие возможности:

Свойства сред: воздух, изотропные и ортотропные материалы с постоянной магнитной проницаемостью, изотропные ферромагнетики, проводники с током, линейные и нелинейные постоянные магниты. Кривые намагничивания ферромагнитных материалов вводятся и редактируются при помощи окна работы с кривыми.

Источники поля: распределенные и сосредоточенные токи или плотность тока, однородное внешнее поле и постоянные магниты.

Граничные условия: заданное значение потенциала (условие Дирихле), заданные значения касательной составляющей индукции (условие Неймана), условие постоянства потенциала (нулевого потока) на поверхностях сверхпроводников.

Результаты расчета: магнитный потенциал, магнитная индукция, напряженность магнитного поля, силы, моменты, энергия магнитного поля, потокосцепления, собственные и взаимные индуктивности.

Специальные возможности: Интегральный калькулятор может вычислять различные интегральные значения на определенных Вами линиях и поверхностях. Магнитные силы могут быть переданы в задачу расчета механических напряжений в элементах конструкции (совмещенная магнито-упругая задача). Мастер индуктивности помогает вычислять собственную и взаимную индуктивность проводников и катушек.

Нестационарное магнитное поле

Данный вид анализа позволяет рассчитывать поле, возбужденное токами произвольной формы и анализировать переходные процессы. Эти задачи возникают при расчете различных машин постоянного и переменного тока, трансформаторов и т. п. В основном, в задачах расчета нестационарного магнитного поля представляет собой интерес определение того, как изменяется во времени магнитная индукция, напряженность поля, токи, силы, моменты, индуктивность и потокосцепление.

Свойства сред: воздух, изотропные и ортотропные материалы с постоянной магнитной проницаемостью, изотропные ферромагнетики, проводники с изменяющимся во времени током, линейные и нелинейные

постоянные магниты. Кривые намагничивания ферромагнитных материалов вводятся и редактируются при помощи окна работы с кривыми.

Источники поля: распределенные и сосредоточенные токи или плотность тока, однородное внешнее поле и постоянные магниты.

В ELCUT есть возможность описывать временные зависимости с помощью формул, используя набор встроенных функций.

Граничные условия: заданное значение потенциала (условие Дирихле), заданные значения касательной составляющей индукции (условие Неймана), условие постоянства потенциала (нулевого потока) на поверхностях сверхпроводников.

Результаты расчета: магнитный потенциал, магнитная индукция, напряженность магнитного поля, силы, моменты, энергия магнитного поля, потокосцепления, собственные и взаимные индуктивности.

Специальные возможности: редактор формул, новое мощное средство, позволяющее описывать практически любой вид источника во времени (ток и плотность тока, условие Неймана). Интегральный калькулятор может вычислять различные интегральные значения на определенных Вами линиях и поверхностях. Магнитные силы могут быть переданы в задачу расчета механических напряжений в элементах конструкции (совмещенная магнито-упругая задача). Омические потери могут быть использованы в качестве источников при расчете нестационарного теплового поля (совмещенная термоэлектрическая задача). ELCUT имеет специальный тип связи двух магнитных задач для передачи начальных условий в нестационарную задачу.

Магнитное поле переменных токов

Данный вид анализа используется для расчета магнитных полей, возбужденных токами, синусоидально изменяющимися во времени и, наоборот, для расчета токов, индуцированных переменным магнитным полем в проводящей среде (вихревых токов). Эти задачи возникают при расчете различных индукторов (в том числе систем индукционного нагрева), соленоидов, электрических машин, и других устройств. Обычно при расчетах магнитного поля переменных токов представляют интерес такие величины, как полный электрический ток (с его сторонней и вихревой компонентами), электрическое напряжение, мощность тепловыделения (омические потери), индукция магнитного поля, напряженность магнитного поля, магнитные силы и их моменты, комплексное сопротивление (импеданс), индуктивность.

При постановке задачи Вы можете использовать следующие возможности:

Свойства сред: воздух, ортотропные материалы с постоянной магнитной проницаемостью, токонесущие проводники с известным напряжением или током.

Источники поля: приложенное напряжение, полный ток проводника, плотность тока или однородное внешнее поле.

Граничные условия: заданное значение потенциала (условие Дирихле), заданные значения касательной составляющей индукции (условие Неймана),

условие постоянства потенциала (нулевого потока) на поверхностях сверхпроводников.

Результаты расчета: векторный магнитный потенциал, плотность тока, напряжение, магнитная индукция, напряженность магнитного поля, силы, моменты, омические потери, вектор Пойнтинга, энергия магнитного поля, импеданс, собственные и взаимные индуктивности.

Специальные возможности: Интегральный калькулятор может вычислять различные интегральные значения на проведенных Вами линиях и поверхностях. Магнитные силы могут быть переданы в задачу расчета механических напряжений в элементах конструкции (совмещенная магнито-упругая задача); а омические потери могут быть использованы в качестве источников тепла при анализе теплового поля (совмещенная термоэлектрическая задача). Два мастера помогают вычислить собственную и взаимную индуктивность катушек и импеданс проводников (полное комплексное сопротивление переменному току).

Электростатика

Расчеты электростатического поля используются при проектировании и исследовании высоковольтного оборудования (разрядников, выключателей, элементов линий электропередачи), изоляционных конструкций, кабелей, конденсаторов, а также при анализе распространения TEM-волн в волноводах. Обычно представляют интерес следующие физические величины: электрический потенциал, напряженность поля, электростатическое смещение (индукция), заряд, емкость и электростатическая сила.

ELCUT может применяться для анализа линейных электростатических полей в плоской и осесимметричной постановках. Задача формулируется в виде уравнения Пуассона относительно электрического потенциала. При постановке задачи Вы можете использовать следующие возможности:

Свойства сред: воздух, изотропные и ортотропные материалы с постоянной диэлектрической проницаемостью.

Источники поля: электроды с заданным потенциалом, распределенные и точечные заряды.

Граничные условия: заданное значение потенциала (условие Дирихле), заданные значения нормальной составляющей поля (условие Неймана), условие постоянства потенциала на поверхностях изолированных проводников.

Результаты расчета: потенциал, напряженность поля, электрическое смещение (индукция), заряд, собственные и взаимные частичные емкости, силы, моменты, энергия: электрического поля.

Специальные возможности: Интегральный калькулятор может вычислять различные интегральные значения на определенных Вами линиях и поверхностях. В задачу могут быть включены изолированные проводники с заранее неизвестным потенциалом (электростатические экраны). Электрические силы могут быть переданы в задачу расчета механических напряжений в элементах конструкции (совмещенная электро-упругая задача).

Мастер емкости поможет Вам вычислить собственную и взаимную емкость проводников.

Электрическое поле постоянных токов

Задача расчета электрического поля постоянных токов используется при анализе различных массивных проводящих систем и при расчете сопротивления заземления (утечки). Величины, представляющие интерес при анализе, включают напряжение, плотность тока, мощность тепловыделения (джоулевы потери).

Задача может быть решена в линейной плоской или осесимметричной постановке. Формулировка задачи основана на уравнении Пуассона для электрического потенциала. При постановке задачи Вы можете использовать следующие возможности:

Свойства сред: воздух, изотропные и ортотропные материалы с постоянной электропроводностью.

Источники поля: электроды с заданным потенциалом, сторонние токоподводы.

Граничные условия: заданное значение потенциала (условие Дирихле), заданные значения нормальной составляющей плотности тока (условие Неймана), условие постоянства потенциала на поверхностях хорошо проводящих включений.

Результаты расчета: потенциал, напряженность поля, плотность тока, ток через заданную поверхность, мощность тепловыделения (джоулевых потерь).

Специальные возможности: Интегральный калькулятор может вычислять различные интегральные значения на определенных Вами линиях и поверхностях. Мощность тепловыделения может быть передана в качестве источника тепла в задачу расчета температурного поля (совмещенная электро-тепловая задача).

Электрическое поле переменных токов

Задача расчета электрического поля переменных токов используется при анализе электрических полей, вызванных переменными токами и напряжениями в неидеальных диэлектриках. Этот вид анализа чаще всего применяется при расчете сложных систем изоляции и конденсаторов. Обычно интерес представляют омические потери в диэлектриках, напряжение, компоненты электрического поля, силы, вращающие моменты.

При постановке задачи Вы можете использовать следующие возможности:

Свойства сред: воздух, ортотропные материалы с постоянной электропроводностью и диэлектрической проницаемостью.

Граничные условия: заданное значение потенциала (условие Дирихле), заданные значения поверхностной плотности тока (условие Неймана), условие постоянного заранее неизвестного потенциала на поверхностях проводников. Результаты расчета: потенциал, компоненты электрического поля, плотность тока проводимости и смещения, мощность омических потерь в диэлектрике и реактивная мощность, силы и вращающие моменты.

Специальные возможности: Интегральный калькулятор может вычислять различные интегральные значения на определенных Вами линиях и поверхностях. Электрические силы могут быть переданы в задачу расчета механических напряжений (совмещенная электро-упругая задача), а электрические потери могут использоваться в качестве источника тепла в задачах теплопередачи (совмещенная электро-тепловая задача).

Теплопередача

Температурный анализ играет заметную роль при проектировании многих механических и электромагнитных систем. Как правило, интерес представляют распределение температуры, температурного градиента, теплового потока и потерь тепла. Используя модуль нестационарной теплопередачи, можно рассчитать тепловой переходный процесс с постоянными во времени граничными условиями.

ELCUT может выполнять линейный и нелинейный стационарный температурный анализ в плоской и осесимметричной постановке. Формулировка задачи основывается на стационарном уравнении теплопроводности с граничными условиями радиационного и конвективного теплообмена. При постановке задачи Вы можете использовать следующие возможности:

Свойства сред: ортотропные материалы с постоянной теплопроводностью, изотропные материалы с теплопроводностью, зависящей от температуры, материалы с теплоемкостью, зависящей от температуры.

Источники поля: постоянные и зависящие от температуры объемные источники тепловой мощности, конвективные и радиационные источники, мощность джоулевых потерь, импортированная из задачи растекания токов.

Граничные условия: заданная температура, заданный тепловой поток на границе, условия радиационного и конвективного теплообмена, а также поверхности с постоянной, заранее неизвестной температурой.

Результаты расчета: температура, градиент температуры, плотность теплового потока и интегральные значения теплового потока через заданные поверхности. Для нестационарной задачи: графики и таблицы изменения физической величины в зависимости от времени.

Специальные возможности: Интегральный калькулятор может вычислять различные интегральные значения на определенных Вами линиях и поверхностях. Распределение температуры может быть передано в задачу расчета механического напряженного состояния (совмещенная термо-упругая задача).

Можно передать распределение температуры в задачу нестационарной теплопередачи, где оно будет использовано в качестве начального распределения температуры.

Задачи теории упругости

Расчет упруго-напряженного состояния применяется при проектировании большинства электрических или механических конструкций. В задачах теории упругости интерес представляют перемещения, деформации и различные компоненты тензора напряжений.

Пакет ELCUT может решать линейную задачу упругости сплошной среды для плосконапряженного, плоскодеформированного или осесимметричного напряженного состояния. Уравнения записаны в терминах напряжений. При постановке задачи Вы можете использовать следующие возможности:

Свойства сред: ортотропные и изотропные материалы.

Источники поля: сосредоточенные силы, распределенные поверхностные и объемные силы, внешнее давление, термические деформации и силы, импортированные из задач электростатики, электрического поля переменных токов и магнитостатики.

Граничные условия: жесткое закрепление с заранее заданным смещением, упругое закрепление (пружинный подвес).

Результаты расчета: перемещения, деформации, координатные и главные компоненты тензора напряжений, критерии Мизеса, Треска, Мора-Кулона, Друкера-Прагера, Хилла.

6. Создание виртуальных панелей управления (LabView)

LabView – универсальная среда для разработки систем сбора, обработки данных и управления экспериментом, включающая обширную библиотеку функций, методов анализа (спектральный и корреляционный анализ, вейвлетный анализ, методы фильтрации, статистическая обработка), библиотеки драйверов устройств. Основой среды является графическое программирование на простом и наглядном языке блок-диаграмм, состоящих из функциональных узлов и связей между ними [11].

Внешний вид и функциональность LabView повторяет традиционный физический прибор, такой как осциллограф или мультиметр. В LabView проектируется лицевая панель виртуального прибора из набора различных управляющих элементов и индикаторов. После создания лицевой панели определяется функциональность виртуального прибора путем набора блок-схемы.

LabView удобно использовать с различными аппаратными средствами, встраиваемыми в персональный компьютер или подключаемыми к нему.

LabView – программная среда, использующая язык графического программирования G. На языке G программа представляется не в виде семантического текстового описания, а в виде графического представления основных операторов программирования на блок-диаграмме и проводников потока данных, определяющих последовательность выполнения программы.

В **LabView** интерфейс пользователя создается с использованием **палитры Элементов** (Controls). Для соблюдения формальной аналогии с реальным прибором интерфейс пользователя называется лицевой панелью. Обычно лицевая панель имеет серый фон рабочего пространства. Собственно программирование осуществляется на панели диаграмм с использованием графических представлений узлов и функций. Исходный код на языке G называется блок-диаграммой. Программист использует для создания блок-диаграммы **палитру Функций** (Function), из которой извлекает, по мере необходимости, графические представления функций [12].

Программная среда **LabView** полностью поддерживает аппаратные средства, имеющие аппаратные интерфейсы типа GPIB, VXI, PXI, RS-232, RS-485, а также DAQ устройства. **LabView** также поддерживает сетевые протоколы TCP/IP и протоколы обмена данными между программами ActiveX.

С помощью программной среды LabView можно разрабатывать программно-аппаратные комплексы для тестирования, измерения, ввода данных, анализа и управления внешним оборудованием. **LabView** – это 32-х разрядный компилятор, который создает как автономные модули (.EXE), так и совместно используемые динамические библиотеки (.DLL).

6.1. Организация программной среды LabView

Диалоговое окно **LabView** содержит следующие компоненты:

1. Кнопка **New VI** – создание нового ВП. Стрелка рядом с кнопкой используется для создания другого типа объектов, например, элемент управления.
2. Кнопка **Open VI** – открытие созданного ранее ВП. Стрелка рядом с кнопкой предназначена для открытия недавно использовавшегося ВП.
3. Кнопка **DAQ Solutions** – запуск мастера решений DAQ Solution Wizard для быстрой настройки аппаратных средств DAQ.
4. Кнопка **Find Examples** – выводит на экран диалоговое окно поиска примеров ВП, систематизированных по категориям.
5. Кнопка **LabView Tutorial** – открывает справочное пособие по использованию LabView. Эта обучающая программа представляет основные концепции LabView.
6. Кнопка **Exit** – закрывает LabView. (MacOS) Кнопка Quit.
7. Секция **Quick Tip** – показывает советы по использованию LabView.
8. Кнопка **Next** – переход к следующему совету.

6.2. Виртуальные приборы (ВП)

Программа, написанная в среде **LabView**, называется виртуальным прибором (ВП). ВП содержит четыре основных компонента:

1. лицевую панель;
2. блок-диаграмму;
3. пиктограмму;
4. соединительную панель (область полей ввода/вывода данных).

Лицевая панель является интерфейсом с пользователем, блок-диаграмма – исходным кодом ВП, а пиктограмма и коннектор – интерфейсом вызова функций. Блок-диаграмма включает в себя ввод/вывод данных, вычисления, и subVI (виртуальные приборы нижнего уровня), которые представлены пиктограммами и связаны линиями, определяющими потоки данных. Компоненты ввода/вывода взаимодействуют непосредственно с встраиваемыми платами сбора данных или интерфейсов КОП (GPIB, IEEE-488.2), VME/VXI, а также с другими внешними физическими приборами. Вычислительные компоненты выполняют арифметические и другие операции. SubVI вызываются другими виртуальными приборами, посылая и получая данные через свои пиктограммы и коннекторы.

6.3. Лицевая панель

Лицевая панель служит интерактивным интерфейсом для обеспечения ввода данных и вывода результатов в вашей измерительной системе.

Когда ваш VI будет закончен, вы используете лицевую панель для управления вашей системой посредством нажатия кнопок, перемещением рукояток, переключателей и вводом информации с клавиатуры. Немедленная реакция системы обеспечивает обратную связь в реальном времени.

Лицевая панель создается с использованием палитры Элементов (Controls). Эти элементы могут быть либо средствами ввода данных – элементы управления, либо средствами отображения данных – элементы отображения. Элементы Управления – кнопки, ручки управления, ползунки и другие элементы ввода. Элементы Отображения – графики, цифровые табло и т. д. После помещения элементов Управления или Отображения данных на Лицевую панель, они получают свое графическое отображение на блок-диаграмме. Элемент, созданный на лицевой панели, невозможно удалить на блок-диаграмме. Объекты блок-диаграммы включают графическое отображение элементов Лицевой панели, операторов, функций, подпрограмм ВП, констант, структур и проводников данных, по которым производится передача данных между объектами блок-диаграммы.

6.4. Блок-диаграмма

Блок-диаграмма является исходным текстом виртуального инструмента. Свободные от множества синтаксических деталей обычного программирования, вы строите блок-диаграмму, выбирая функциональные блоки из палитр меню Functions. Затем – соединяете блоки линиями, называемыми проводниками (wires), для прохождения данных от одного блока к другому. Эти блоки могут быть простыми арифметическими функциями, виртуальными инструментами для сбора данных и анализа, сетевыми функциями и функциями ввода/вывода, которые записывают и восстанавливают данные в бинарном, ASCII или даже в табличном виде.

6.5. Поточное программирование

Функциональные пиктограммы и структуры часто рассматриваются как узлы (nodes).

Программа **LabView** выполняется в соответствии со следующими принципами:

1. Узел выполняется только тогда, когда на все его входы поступили данные.
2. Узел выдает данные на все выходы только тогда, когда заканчивает выполняться заложенный в нем алгоритм.
3. Данные передаются от источника к приемнику без задержки.

Узлы – это выполнимые элементы программы. Они аналогичны инструкциям, операторам, функциям, и подпрограммам в стандартных языках программирования. **LabView** имеет мощную библиотеку функций для математических вычислений, сравнений, преобразований, ввода/вывода и других действий.

Другой тип узлов – структура. Структуры – это графические представления циклов и операторов выбора традиционных языков программирования, которые повторяют блоки исходного текста или выполняют их в зависимости от условия. **LabView** также имеет специальные узлы для компоновки с внешним текстово-основанным кодом и для обработки текстово-основанных формул.

Провода – линии данных между источником и приемником. Вы не можете присоединить пиктограмму выхода к другой пиктограмме выхода, или

пиктограмму входа к пиктограмме входа. Вы имеете возможность присоединять один источник к нескольким приемникам. Каждый провод имеет различный вид или цвет, в зависимости от типа данных, которые передаются по этому проводу.

Поточное программирование освобождает вас от линейной архитектуры текстовых языков. Так как порядок выполнения программы в **LabView** устанавливается течением данных между узлами, а не последовательностью строк текста, вы можете создавать диаграммы, которые имеют несколько параллельных потоков прохождения данных и одновременных операций. **LabView** обеспечивает прохождение нескольких потоков данных и выполнение независимых блоков одновременно.

6.6. Графический компилятор

Во многих приложениях скорость выполнения является критичной. **LabView** – единственная графическая среда программирования с компилятором, который генерирует оптимизированный код. Скорость выполнения **LabView** близка к скорости выполнения компилированных Си программ.

Вы можете создавать виртуальные приборы и запускать их в **LabView Run-Time System**. Это компактная дешевая версия **LabView** может только загружать и запускать ВП, но не может редактировать или показывать их диаграмму. Вы можете использовать **Run-Time System** как дешевую тестовую станцию или эффективный путь для распространения собственных разработок.

Кроме того, с помощью дополнительной программы **Application Builder** выполняется преобразование ВП в обычную исполняемую *.exe программу, которая запускается и выполняется самостоятельно, как любая Windows программа.

LabView – открытая система, поэтому вы можете включать в систему свои собственные программные и аппаратные разработки. Для включения объектного Си кода в **LabView** программу вы можете воспользоваться 32-х разрядным компилятором Watcom C.

6.7. Модульность и иерархия

Преимущество **LabView** заключается в иерархической структуре ВП. Созданный виртуальный прибор можно использовать в качестве подпрограммы на блок-диаграмме ВП более высокого уровня. Количество уровней в иерархии не ограничено. Использование подпрограммы ВП помогает быстро изменять и отлаживать блок-диаграмму.

LabView является модульной средой по своей структуре. То есть, любой ВП может использоваться в блок-диаграмме другого виртуального прибора как subVI. Разбив свою программную систему на subVI, вы можете независимо разработать и интерактивно протестировать эти subVI, и тут же использовать их как узлы для построения более сложного уровня ВП. Использование модульной иерархии позволяет эффективно разрабатывать, модифицировать,

заменять и комбинировать виртуальные инструменты для удовлетворения изменяющихся требований конкретного приложения.

Кроме того, ваши возможности значительно расширяет иерархия ВП. Создавая пиктограмму для собственного ВП и используя ее в диаграмме другого виртуального инструмента, вы скрываете сложность низкоуровневой диаграммы, однако сохраняете доступ к общим переменным через панели нижнего уровня. Вы можете даже конфигурировать эти панели для автоматического открытия, создания анимаций и интерфейса пользователя.

7. Интегрированная среда разработки ATMEL AVR Studio

AVR Studio – это интегрированная отладочная среда разработки приложений для микроконтроллеров AVR компании Atmel.

AVR Studio содержит :

1. средства создания и управления проектом;
2. редактор кода на языке ассемблер;
3. транслятор языка ассемблера (Atmel AVR macroassembler);
4. отладчик (Debugger);
5. программное обеспечение верхнего уровня для поддержки внутрисхемного программирования (In-System Programming, ISP) с использованием стандартных отладочных средств Atmel AVR.

Работа с AVR Studio начинается с создания проекта. При создании проекта необходимо указать используемый микроконтроллер и платформу, на которой будет производиться отладка программы [13].

Написание программы производится в окне редактора текста программы. Для использования символических имен регистров специального назначения вместо их адресов необходимо подключить (директива .include) к проекту файл определения регистров специального назначения (например, m16def.inc для ATmega16). Включаемые файлы входят в прикладное программное обеспечение AVR Studio и при инсталляции помещаются в папку Appnotes в директории установки AVR Studio. Примеры программ доступны в большом количестве в качестве приложений к руководствам по применению микроконтроллеров AVR.

Написание программы в AVR Studio производится на языке ассемблер. Система команд микроконтроллера описана в упомянутых выше руководствах по применению в документе AVR Instruction Set либо в файле справки, встроенном в AVR Studio (меню Help \ AVR Tools User Guide \ AVR Assembler), в котором содержатся достаточно подробные комментарии к каждой команде.

Последние версии AVR Studio содержат тестовую версию AVR ассемблера второй версии, который в дополнение к стандартному ассемблеру поддерживает новые директивы ассемблера, Си - подобные директивы препроцессора, создание переменных определенного типа. Более подробную информацию можно найти в файле справки.

Перед трансляцией программы можно задать установки проекта (меню Project \ AVR Assembler Setup), указать необходимый формат выходного файла. Там же возможно установить использование AVR ассемблера 2-ой версии. Так как вторая версия ассемблера проходит стадию тестирования, то по умолчанию он отключен. Если не требуется каких-либо особых настроек, то можно использовать установки по умолчанию.

В результате трансляции создается выходной файл в указанном формате. Если исходный ассемблерный текст содержал сегмент энергонезависимых данных (объявленный директивой .eseg), то при трансляции будет создан также файл с расширением .eep. Этот файл содержит данные для внутренней EEPROM микроконтроллера и имеет тот же формат, что и выходной файл.

Если в результате трансляции не выдается сообщений об ошибках, можно приступать к отладке проекта.

Отладчик AVR Studio поддерживает все типы микроконтроллеров AVR и имеет два режима работы: режим программной симуляции и режим управления различными типами внутрисхемных эмуляторов (In-Circuit Emulators) производства фирмы Atmel. Важно отметить, что интерфейс пользователя не изменяется в зависимости от выбранного режима отладки.

Отладочная среда поддерживает выполнение программ, как в виде ассемблерного текста, так и в виде исходного текста языка Си, загруженного в объектном коде.

После загрузки объектного кода производится выбор отладочной платформы и используемого микроконтроллера. При отладке с использованием внутрисхемных эмуляторов, микроконтроллеры, поддержка которых не осуществляется, автоматически подсвечиваются серым. После загрузки проекта система готова к старту отладки [14].

Управление отладкой производится командами меню DEBUG, либо соответствующими иконками на панели управления AVR Studio.

Пользователь может выполнять программу полностью в пошаговом режиме, трассируя блоки функций, или выполняя программу до того места, где стоит курсор. В дополнение можно определять неограниченное число точек останова, каждая из которых может быть включена или выключена. Точки останова сохраняются между сессиями работы.

В AVR Studio для отладки программы предусмотрены две команды пошагового режима: Step Over и Trace into. Разница между ними в том, что при выполнении команды Step Over выполнение подпрограмм происходит до полного окончания без отображения процесса выполнения. К командам шагового режима также относится команда Auto Step.

С помощью команд пошагового режима можно проследить изменения значений в переменных, регистрах ввода/вывода, памяти и регистрового файла. Для этого предназначены раздел I/O рабочей области AVR Studio (см. картинку), окно Watch (меню Debug \ Quickwatch).

Интегрированная среда разработки AVR Studio также поддерживает просмотр (меню View \ Memory) ячеек памяти программ, памяти данных, EEPROM и регистров портов ввода/вывода в ходе исполнения. Выпадающее меню диалогового окна позволяет выбрать один из четырех массивов ячеек памяти: Data, IO, Eeprom, Program Memory. Для одновременного просмотра нескольких областей окно Memory может быть открыто несколько раз. Информация в диалоговом окне может быть представлена в виде байтов или в виде слов в шестнадцатеричной системе счисления, а также в виде ASCII - символов.

В процессе отладки пользователь может инициализировать внутреннее ОЗУ или EEPROM микроконтроллера (например, данными, содержащимися в полученном при трансляции файле .eep), или сохранить содержимое ОЗУ и EEPROM в виде файлов в формате Intel Hex (меню File -> Up/Download Memory).

Помимо шагового режима, возможна отладка программы с использованием точек останова (меню Breakpoints -> Toggle Breakpoint). Командой Start Debugging запускается исполнение программы. Программа будет выполняться до остановки пользователем или до обнаружения точки останова.

Работая с программным симулятором пакета AVR Studio, следует помнить, что он пока не поддерживает некоторые режимы работы микроконтроллеров AVR и их периферийные узлы:

1. Аналого-цифровой преобразователь.
2. Аналоговый компаратор.
3. Режим часов реального времени.
4. Режим пониженного энергопотребления (инструкция «sleep» интерпретируется программным симулятором как «nop»).

Для наблюдения за работой программы можно открыть несколько окон, отображающих состояние различных узлов микроконтроллера (см. рисунок). Окна открываются нажатием соответствующих кнопок на панели инструментов или при выборе соответствующего пункта меню View. Если в процессе выполнения программы в очередном цикле значение какого-либо регистра изменится, то этот регистр будет выделен красным цветом. При этом, если в следующем цикле значение регистра останется прежним, то цветовое выделение будет снято. Такое же цветовое выделение реализовано в окнах устройств ввода/вывода, памяти и переменных.

Состояние встроенных периферийных устройств микроконтроллера, а также состояния программного счетчика, указателя стека, содержимого регистра статуса SREG и индексных регистров X, Y и Z отображено в окне I/O Window. В этом окне отражаются все функциональные блоки микроконтроллера. Любой блок может быть раскрыт нажатием на его значок. При раскрытии блока в окне отражаются адреса и состояния всех его регистров и отдельных, доступных для модификации, битов. Каждый доступный для модификации бит может быть установлен или сброшен как программой по ходу ее исполнения, так и пользователем вручную (указав курсором мыши нужный бит и, щелкнув левой кнопкой мыши, пользователь может изменить значение бита на обратное), а в режиме программной симуляции это является способом имитации входного воздействия на микроконтроллер [13, 14].

Другим способом задания входного воздействия на микроконтроллер в режиме симулятора является использование внешних файлов входных воздействий. Формат файла входного воздействия очень прост:

```
000000000:00
000000039:01
000000040:00
9999999999:FF
```

Здесь значение, указанное после разделителя « : » – это шестнадцатеричное представление сигналов, воздействующих на порт микроконтроллера. Значение, указанное до разделителя – это десятичный номер цикла (с момента сброса микроконтроллера), в котором указанное воздействие поступает на выводы порта микроконтроллера. Файл входного воздействия должен

заканчиваться строкой с заведомо большим номером цикла, в противном случае будет выдано сообщение об ошибке. Для подключения файла входного воздействия служит пункт меню Debug \ Options в разделе Stimuli and Logging. В открывшемся окне нужно указать порт микроконтроллера, на который нужно подавать воздействие, и файл этого воздействия. Пользователь может создавать файлы воздействий, а также записывать изменения значений на выходах портов микроконтроллера в файл (формат этого файла тот же, что и у файла входных воздействий).

Для записи служит функция Logging. В открывшемся окне нужно указать порт микроконтроллера и имя файла для записи. Записываемый файл будет удаляться и создаваться вновь при каждом выполнении сброса микроконтроллера (Debug > Reset). Подключать файл входного воздействия или задавать имя файла для записи пользователь должен сам при каждом запуске симулятора.

Как Вы уже, наверное, заметили, в тексте статьи не раз были приведены ссылки на встроенные файлы справки AVR Studio. Следует отметить, что AVR Studio имеет очень мощную встроенную документацию как по использованию AVR Studio, так и по использованию стандартных отладочных средств производства компании Atmel, а также по системе команд и всему, что касается программирования с использованием AVR Studio. Таким образом, использование встроенной справочной информации избавляет разработчика от накопления разобщенной информации, учитывая, что встроенная информация обновляется с выходом новых версий AVR Studio.

8. Сетевые технологии в электроприводе

Говоря о роли электропривода в автоматизированных распределенных системах, следует иметь ввиду прежде всего системы АСУТП, поскольку именно в них ЭП является основным средством осуществления необходимых операций по приведению в движение разнообразных рабочих органов технологического оборудования и обрабатываемых материалов.

Автоматизированная система управления технологическими процессами – совокупность аппаратно-программных средств, осуществляющих контроль и управление производственными и технологическими процессами, поддерживающих обратную связь и активно воздействующих на ход процесса при отклонении его от заданных параметров, обеспечивающих регулирование и оптимизацию управляемого процесса.

ЭП успешно применяется не только в качестве основного источника энергии, с помощью которой происходит механическая обработка, но выполняет также множество вспомогательных функций по требуемому перемещению рабочих органов, осуществляющих другие виды обработки (например, лазерная обработка, покраска, производство печатной продукции и т. п.). Не очень заметны, но, безусловно, необходимы ЭП многочисленных насосных и вентиляционных установок, которые являются первичными источниками энергии для пневмо- и гидроагрегатов, поддерживают требуемые условия окружающей среды в промышленных, административных и жилых помещениях. Дистанционное управление различными заслонками, задвижками, вентилями также осуществляется исполнительными механизмами на базе ЭП.

Постоянно расширяется сфера применения регулируемых ЭП, с помощью которых удается повысить качество выпускаемой продукции, экономить материальные и энергетические ресурсы. Переход от нерегулируемого к регулируемому ЭП, как правило, увеличивает объем информационного потока между ЭП, оператором и внешними управляющими устройствами. Наличие в силовом канале регулируемого ЭП электронного преобразователя, управляемого достаточно мощным микроконтроллером, создает условия для более качественного управления, контроля и диагностики как собственно ЭП, так и технологического оборудования.

Развитие систем управления (СУ) ЭП происходит как в части совершенствования алгоритмов регулирования координат электро-механического преобразователя (токи, момент, скорость), так и за счет расширения вспомогательных функций по защите, диагностике, автоматической настройке регуляторов. Обеспечивается несколько уровней человеко-машинного интерфейса, позволяющих не только контролировать основные параметры работы ЭП, но и осуществлять протоколирование аварийных ситуаций, визуализацию процессов, дистанционный контроль и управление.

Наблюдается тенденция встраивания в СУ ЭП все более развитых микропроцессорных средств, способных выполнять функции управления, возлагаемые обычно на ПЛК и системы ЧПУ – формирование требуемых

законов изменения задающих воздействий на входы регуляторов координат электропривода. Появляется возможность реализации в той или иной мере функций технологических контроллеров – регуляторов технологических величин. И, наконец, СУ ЭП оснащаются встроенными средствами для обмена данными с внешними устройствами по промышленным коммуникационным сетям, что позволяет интегрировать современный ЭП в распределенные системы АСУТП. Можно говорить, что ЭП становится интеллектуальным устройством в составе АСУТП, совмещающим множество различных функций.

В классификации автоматизированных систем производства ЭП является элементом нижних уровней и, следовательно, может взаимодействовать как непосредственно с датчиками различных технологических величин, так и с другими ЭП, исполнительными механизмами и системами, чем обеспечивается согласованная работа всех составных частей технологической установки.

Для нижнего уровня производственных систем характерны следующие общие типовые задачи:

- хранение и выполнение общего алгоритма управления;
- локальное аналоговое управление;
- локальное цифровое управление;
- связь с оператором;
- связь с компьютером верхнего уровня.

Наиболее сложными и многообразными являются задачи аналогового управления непрерывной частью объекта, которые удобно разделить на четыре уровня.

Первый уровень – задачи сбора, первичной обработки и преобразования информации, т. е. предварительная аналоговая или цифровая фильтрация сигналов, нормализация и масштабирование, аналого-цифровое преобразование, функциональные преобразования, вычисление спектральных характеристик сигналов и т. п. Важность задач этого уровня определяется еще и тем обстоятельством, что именно от эффективного решения задач на этом уровне зависит сохранение точности дальнейших преобразований и обработки, погрешности измерения могут существенно исказить результаты вычислений. Одной из устойчивых тенденций является приближение обработки информации к точкам ее получения.

Алгоритмы решения задач на этом уровне (особенно связанные с цифровой фильтрацией и спектральной обработкой) предъявляют очень высокие требования к вычислительным средствам, в части требуемых объемов памяти и быстродействия.

Второй уровень – задачи геометрических преобразований систем координат, связанных с различными функциональными пространствами; определения текущих параметров измеренных процессов, представленных в цифровой форме (например, скоростей и ускорений его протекания); совместного решения линейных и нелинейных алгебраических уравнений, в том числе и рекуррентного типа; решения систем дифференциальных уравнений (в том числе в частных производных); прогнозирования хода

управляемых процессов; интерполирования и экстраполирования вычисляемых функций; поиска данных в таблицах; цифроаналоговых преобразований и т. п. Часто совокупность задач, решаемых на этом уровне, характеризуют одним термином – вторичная обработка информации.

Третий уровень – задачи оптимизации, т. е. задачи поиска экстремума функции (функционала) одной или нескольких переменных, на которые наложены определенные ограничения. Методы решения таких задач составляют предмет раздела прикладной математики, называемого математическим программированием. В зависимости от характера целевой функции (критерия оптимизации) и ограничений, наложенных на управляемые переменные, эти задачи решаются методами линейного, нелинейного, стохастического, целочисленного и динамического программирования. Решение этих задач, как правило, связано с предъявлением высоких требований к вычислительным ресурсам, особенно для задач большой размерности.

Четвертый уровень – задачи представления информации в виде, удобном для восприятия человеком – оператором. На основе этой информации он может либо принимать решение по введению в систему управляющих воздействий, либо наблюдать за правильностью функционирования технических средств, составляющих систему или системы. Характер используемых методов решения таких задач зависит от конкретного состава и принципов организации устройств отображения информации и конструкции органов управления.

Не все из указанных задач целесообразно решать встроенными микропроцессорными средствами ЭП. Многие производители комплектных ЭП выпускают специальные опции, предназначенные для автономной работы небольших установок и совмещающие в себе функции ПЛК, контроллера управления движением и собственно системы управления ЭП.

Заключение

Представленный конспект лекций посвящен курсу «Компьютерные технологии в электроприводе». Дается структура компьютерных технологий по их использованию для различных технических задач в электроприводе.

В конспекте лекций приведены описания современных программ, которые применяются при проектировании и эксплуатации электропривода. В некоторых случаях рассматриваются иллюстративные примеры.

Представленные материалы предназначены для студентов специальности «Электропривод и автоматизация промышленных установок и комплексов» и могут быть использованы в курсах моделирования, программирования и создания виртуальных панелей управления.

Библиографический список

1. Клиначев Н. В. Моделирование обыкновенных линейных систем: руководство к лабораторным работам в пакетах VisSim и Workbench / Н. В. Клиначев.– Челябинск, 2001. – 35 с.
2. Клиначев Н. В. Моделирование систем в программе VisSim / Н. В. Клиначев. – 2001.
3. OrCAD. Моделирование с помощью Pspice.
4. B2 Spice A/D 4.2. User`s manual: 2003. – 256 с.
5. Начало работы с MATLAB. – 74 с.
6. MATLAB: Simulink & Toolboxes. – 61 с.
7. Черных И. В. Simulink: Инструмент моделирования динамических систем / И. В.Черных.– М.: Диалог-МИФИ, 2003. – 496 с.
8. Бенькович Е. С. Практическое моделирование динамических систем / Е. С. Бенькович, Ю. Б. Колесов, Ю. Б. Сениченков.– СПб. : БХВ-Петербург, 2002. – 464 с.
9. Model Vision Studium 3.0. Руководство пользователя.
10. ELCUT 5.2. Руководство пользователя – СПб. : ПК ТОР, 2005. – 258 с.
11. LabView 7 Express. Вводный курс – М. : изд-во «ПриборКомплект», 2003. – 42 с.
12. LabView 7 Pro. Руководство пользователя.
13. Курилин А. И. AVR – микроконтроллеры: семь ярких лет становления. Что дальше? Часть 3 / А. И. Курилин, Р. Н. Золотухо // Компоненты и технологии. – 2005, №1.
14. Atmel AVR Studio 4. User`s manual.
15. Электропривод и сетевые технологии : доклады научно-практического семинара. – М. : изд-во МЭИ, 2003. – 144 с.