

681.32 (075)

Ш....

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Технологический институт
Федерального государственного образовательного
учреждения высшего профессионального образования
«Южный федеральный университет»
ПРИОРИТЕТНЫЙ НАЦИОНАЛЬНЫЙ ПРОЕКТ «ОБРАЗОВАНИЕ»**

Шеболков В.В.

**МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ КОМПОНЕНТОВ РАДИО-
ЭЛЕКТРОННЫХ УСТРОЙСТВ И СИСТЕМ В ПАКЕТЕ SIMULINK**

Учебное пособие

Часть 1

Таганрог 2007

УДК 621.317

В.В. Шеболков. Моделирование динамических компонентов радиоэлектронных устройств и систем в пакете Simulink. Учебное пособие. - Таганрог: Изд-во ТТИ ЮФУ, 2007.- 104с.

В первой части пособия дается краткая характеристика пакета Simulink, его связи с системой MatL и описывается технология моделирования динамических компонентов радиоэлектронных устройств и систем на системотехническом уровне. Рассмотрены способы построения математических моделей динамических звеньев для систем непрерывного и дискретного времени. Описываются способы перехода от систем непрерывного времени к системам дискретного времени. Подробно описана библиотека компонентов Simulink. Описание всех компонентов проиллюстрировано примерами.

Табл. 3. Илл. 31. Библиогр.; 10 назв.

Рецензенты:

© Технологический институт

Южного федерального университета, 2007

© В.В. Шеболков, 2007

Содержание

Введение.....	4
1. Основы работы в Simulink.....	7
2. Методы построения математических моделей.....	30
2.1. Основные понятия математического моделирования	30
2.2. Методы описания непрерывных линейных систем	33
2.3. Методы описания дискретных линейных систем	48
3. Библиотеки Simulink.....	65
3.1. Commonly Used Blocks - блоки общего применения	66
3.2. Continious - линейные блоки непрерывного времени	71
3.3. Discontinious - нелинейные блоки	73
3.4. Discrete -дискретные блоки	76
3.5. Logic and Bit Operations - логические и битовые операции	79
3.6. Math Operations - математические операции	84
3.7. Signal Routing - блоки маршрутизации сигналов	89
3.8. Sources - источники сигналов	94
3.9. Sinks - приемники сигналов.....	99
3.10. Обзор библиотеки Simulink Extras	102
Библиографический список.....	103

ВВЕДЕНИЕ

Пакет Simulink – это интегрированное в MatlLab ядро интерактивного программного комплекса, предназначенное для математического моделирования линейных и нелинейных динамических систем и устройств, представленных в графической форме: в виде блок-схемы либо в форме графа состояний. Такие представления моделей систем называют S-моделями, а соответствующие им файлы программ S-файлами.

Класс систем, работу которых может моделировать Simulink, чрезвычайно широк: это непрерывные и дискретные системы, это электрические, механические и комбинированные системы, это сосредоточенные и распределенные системы и т.д. – любые динамические системы, работа которых описывается дифференциальными или разностными уравнениями.

Simulink позволяет моделировать работу систем во временной и частотной областях, определять спектры входных воздействий и откликов системы на них, моделировать реакцию систем на воздействия случайного характера и т. д.

Для построения блок-схем моделируемых устройств в Simulink встроена обширная библиотека блочных компонентов и удобный графический редактор блок-схем, т.е. Simulink по существу является типичным средством визуально-ориентированного программирования. Используя палитры (наборы) компонентов, пользователь с помощью мыши переносит нужные функциональные блоки с палитр на рабочий стол графического редактора, задает требуемые параметры и

соединяет линиями входы и выходы блоков. Таким путем создается блок-схема системы или устройства, то есть S-модель.

Simulink автоматизирует наиболее трудоемкий этап моделирования: он составляет и решает сложные системы алгебраических и дифференциальных уравнений, описывающих моделируемую систему или устройство.

Библиотека компонентов (блоков) Simulink открыта для изучения и модификации [1, 2]. Она включает широкий спектр источников сигналов, масштабирующих, линейных и нелинейных преобразователей с разными передаточными функциями и характеристиками, квантующие устройства, виртуальные и т. д. регистрирующие устройства (от простых измерителей типа вольтметра или амперметра до универсальных осциллографов и графопостроителей), средства верификации и оптимизации системы и т.д.

Программные средства моделирования динамических систем известны давно, к ним относятся, например, программы Tutsim и LabVIEW for Industrial Automation [2, 4]. Однако для эффективного применения таких средств необходимы высокоскоростные решающие устройства. Интеграция одной из самых быстрых матричных математических систем — MatLab с пакетом Simulink открывает новые возможности использования современных математических методов для решения задач динамического и ситуационного моделирования сложных систем и устройств. Важным достоинством Simulink является интеграция не только с системой MatLab, и с рядом других пакетов. В последней версии пакета появились средства для его интеграции с системами автоматизированного проектирования цифровых устройств на ПЛИС (программируемых логических интегральных схемах), что существенно расширяет спектр возможностей применения Simulink для решения задач имитационного и событийного моделирования.

Матричная система MatLab реализует решение систем уравнений, описывающих моделируемые объекты, в матричной форме. Все исходные данные и результаты вычислений эта система трактует как векторы. Даже скалярные величины трактуются системой как векторы в од-

номерном пространстве. Наиболее естественной формой описания внутреннего состояния исследуемых систем для MatLab является матричная форма, и все вычисления выполняются по правилам выполнения матричных операций. Simulink, являясь подсистемой MatLab, полностью соответствует этой концепции и часто недоразумения при работе с Simulink происходит вследствие того, что пользователи не учитывают этот фактор.

Процесс моделирования работы исследуемых устройств в Simulink в общем случае включает следующие операции:

1) разработка модели и ее представление в графической форме: либо в виде блок-схемы, либо в форме графа состояний, построенных из компонентов, входящих в библиотеку Simulink;

2) ввод модели в Simulink и установка параметров ее библиотечных компонентов в соответствии с параметрами моделируемого устройства;

3) выбор и установка конфигурационных параметров подсистем Simulink, выполняющих моделирование: алгоритма решения системы уравнений, интервала дискретизации, точности, критериев оптимизации алгоритма, передаваемых в рабочую область MatLab данных и т.д.;


4) пробный запуск и верификация модели;

5) моделирование (Simulation) - исследование работы моделируемого устройства на созданной модели;

6) реализация результатов моделирования.

В учебном пособии рассмотрены особенности и технология выполнения каждого из этих этапов.

1. ОСНОВЫ РАБОТЫ В SIMULINK

1.1. В процессе инсталляции Simulink автоматически интегрируется с MatLab, образуя общую с ним среду. Это выражается и в том, что Simulink запускается из MatLab (либо из командного окна командой Simulink либо нажатием кнопки  на панели инструментов), и в том, что обе программы используют общее рабочее пространство (это позволяет им обмениваться данными), и в том, что запуск, отладку и установку параметров Simulink модели можно выполнять из командного окна MatLab.

Пользовательский интерфейс Simulink вполне традиционен для

Windows-приложений. Его основу составляет окно просмотра библиотек - Simulink Library browser (рис. 1.1), которое появляется на мониторе компьютера после вызова Simulink. В левом поле окна выводится в форме древовидной структуры названия основных разделов и подразделов библиотеки Simulink. Это поле предназначается для выбора нужного раздела или подраздела библиотеки, содержание которых отображается в правом поле в виде графических изображений и названий соответствующих модулей. Знак “+”

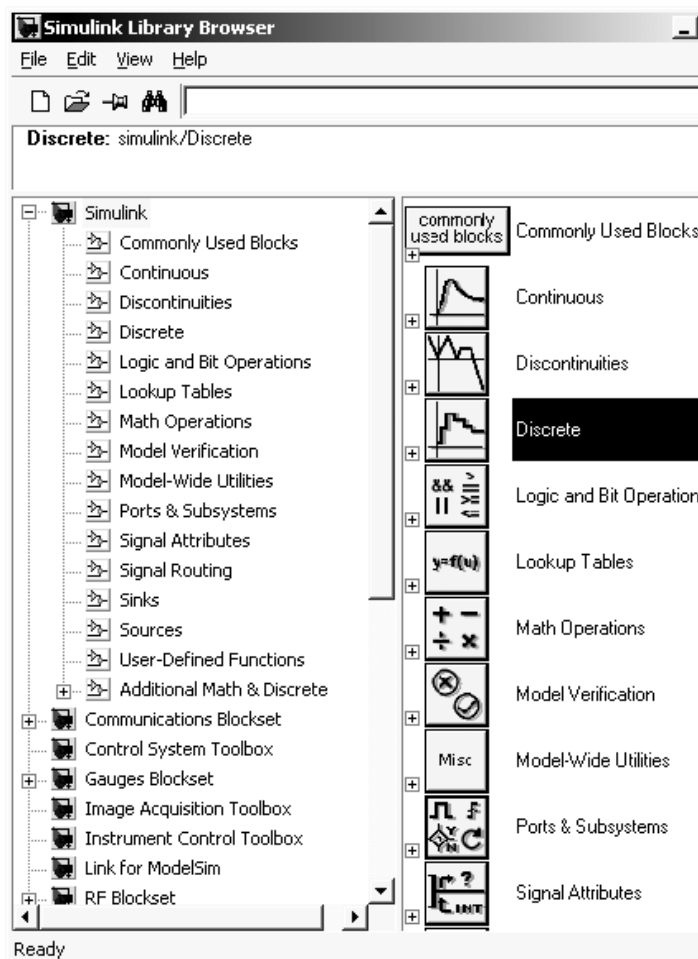


РИС. 1.1

бражения означает, что это изображение является не модулем, а раз-

делом или подразделом библиотеки. Для просмотра его содержимого необходимо щелкнуть один раз левой кнопкой мыши по знаку “+” либо дважды по графическому изображению раздела. После раскрытия дерева выбранного раздела знак “+” слева от заголовка раздела изменяется на “-”. Щелкнув левой кнопкой мыши по нему можно “свернуть” дерево этого раздела. Пояснения к выделенному фрагменту библиотеки (подразделу или модели блока) отображается в текстовой форме под панелью инструментов. Справа от панели инструментов расположено окно для быстрого поиска в библиотеке модели блока по имени.

Основой библиотеки является раздел Simulink, который содержит разделы с наиболее часто употребляемыми блоками для разных предметных областей моделирования. К таким разделам следует отнести, прежде всего, следующие:

- Commonly Used Blocks* - блоки общего назначения,
- Continuous* - линейные аналоговые блоки,
- Discontinuous* - нелинейные аналоговые блоки,
- Discrete* – блоки дискретного времени,
- Logic and Bit Operations* – блоки выполнения логических и битовых операций,
- Math Operations* - блоки выполнения математических операций,
- Sources* – блоки источников сигналов,
- Sinks* – блоки-регистраторы результатов моделирования.

Более подробное описание разделов библиотеки приведено ниже (см. раздел 3). Команды главного меню являются типичными для Windows-приложений. Отсутствие в разделе Edit главного меню команд Copy, Paste, Cut, Select All и т.п. объясняется тем, что графические файлы пользователя (модели и библиотеки) создаются в отдельных рабочих окнах, которые вызываются, соответственно, командами File|New|Model (Ctrl+N) и File|New|Library.

Интерфейс рабочего окна Simulink полностью соответствует по стилю интерфейсу Windows-приложений. Главное меню рабочего окна содержит следующие пункты:

1) File – работа с файлами моделей и библиотек (их создание, сохранение, считывание и печать) и настройка системы “под себя” (Preferences);

2) Edit – редактирование файлов, работа с буфером обмена, создание подсистем;

3) View – управление отображением модели, настройка панели инструментов, быстрый переход в окна MatLab, Simulink Library browser, Model Explorer;

4) Simulation – управление процессом моделирования (старт, пауза, настройки параметров моделирования (Configuration Parameters), выбор режима моделирования;

5) Format – операции форматирования модели (смена шрифтов, редактирование надписей, повороты блоков, использование тени от блоков, операции с цветами блоков, с цветами фона;

6) Tools – инструменты для отладки моделей, RTW), редактирования объектов моделирования, управления видами анализа (в линейной области и в режиме реального времени, записи и диагностики результатов моделирования работы блоков Simulink, кодирование моделей систем в HDL коды для автоматизации проектирования цифровых устройств и т.д.

Первые три пункта главного меню содержат общепринятые для Windows-приложений команды и операции, потому нет необходимости

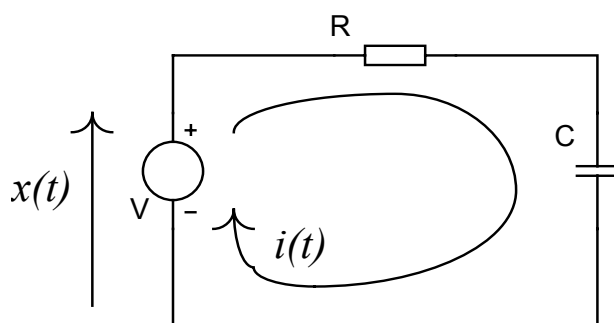


Рис. 1.2

их описывать. Работа с командами и операциями, содержащимися в пунктах Format и Simulation будет проиллюстрирована на примерах. Работа с некоторыми наиболее употребительными инструментами из меню Tools при

исследованиях проектировании компонентов радиоэлектронных устройств, иллюстрируется в разделах 2, 3.

1.2. Чтобы получить представление о возможностях Simulink и принципах работы с ним рассмотрим несколько простых примеров.

Пример 1.1. Моделирование $y(t)$ реакции RC-цепи на воздействие сигнала $x(t)$ на выходе источника напряжения V (рис. 1.2).

В качестве основы для построения модели в Simulink будем использовать дифференциальное уравнение, связывающее напряжение на конденсаторе $y(t)$ и напряжение генератора $x(t)$. Решив это уравнение средствами Simulink, мы решим поставленную задачу.

Поскольку цепь последовательная, то ток во всех элементах цепи одинаков, и его можно выразить через напряжение на конденсаторе $y(t)$:

$$i(t) = i_c(t) = C \frac{dy}{dt}.$$

Тогда в соответствии со вторым законом Кирхгофа

$$Ri(t) + y(t) = x(t).$$

Подставив в это уравнение $i(t)$ из предыдущей формулы, получим искомое дифференциальное уравнение

$$RC \frac{dy}{dt} + y(t) = x(t).$$

Для решения этого уравнения в Simulink его необходимо представить в виде функциональной схемы некоторого абстрактного устройства, составленного из библиотечных модулей Simulink. Чтобы составить такую схему последнее уравнение удобнее представить в несколько ином виде:

$$\frac{dy}{dt} = \frac{1}{RC} (x(t) - y(t)) \quad (1.1)$$

В соответствии с уравнением (1.1) если выходное напряжение интегратора равно $y(t)$, то его входное напряжение равно $\frac{dy}{dt}$ и равно $\frac{1}{RC} (x(t) - y(t))$. Функциональная схема устройства, построенная по

этому уравнению, представлена на рис. 1.3. Она содержит сумматор Σ , перемножитель \times , интегратор \int и блок вычисления величины $\frac{1}{RC}$.

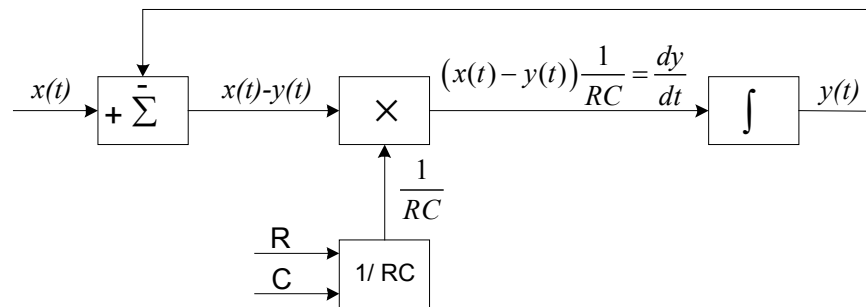


Рис. 1.3

На этом этап подготовки модели можно считать законченным и переходить к вводу модели в Simulink.

Вид рассматриваемой модели в Simulink показан на рис. 1.4. На поле модели приведены выражения для сигналов на входах и выходах каждого блока.

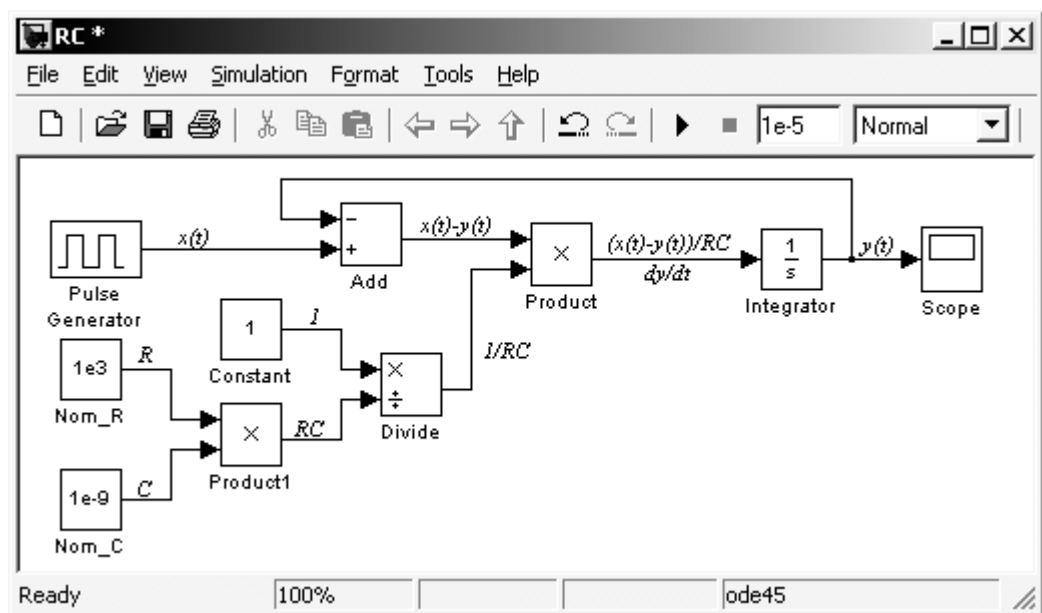


Рис. 1.4

В качестве источника сигнала используется генератор прямоугольных импульсов Pulse Generator. Блоки Constant, Nom_R, Nom_C, Product1 и Devide предназначены для вычисления множителя $1/RC$ в уравнении (1.1). Осциллограф Scope используется в качестве инст-

румента наблюдения результатов моделирования. Назначение остальных блоков соответствует функциональной схеме (рис. 1.3).

Ввод модели в Simulink выполняется следующим образом.

1) Из окна Simulink Library browser открываем рабочее окно для создания новой модели (команда File|New|Model или Ctrl+N). Этому окну Simulink по умолчанию присвоит имя “untitled”, все вносимые в него данные будут сохраняться в файле с именем “untitled.mdl”, который буде помещен в папку ...\\MatLab\\R2006b\\work. Имя файла и место его нахождения пользователь может изменить по своему усмотрению (в имени файла не допускается использовать символы кириллицы!!!). Сохраним пока еще пустой файл под именем Pr_1_1 в папке ...\\MatLab\\R2006b\\work, предложенной системой по умолчанию.

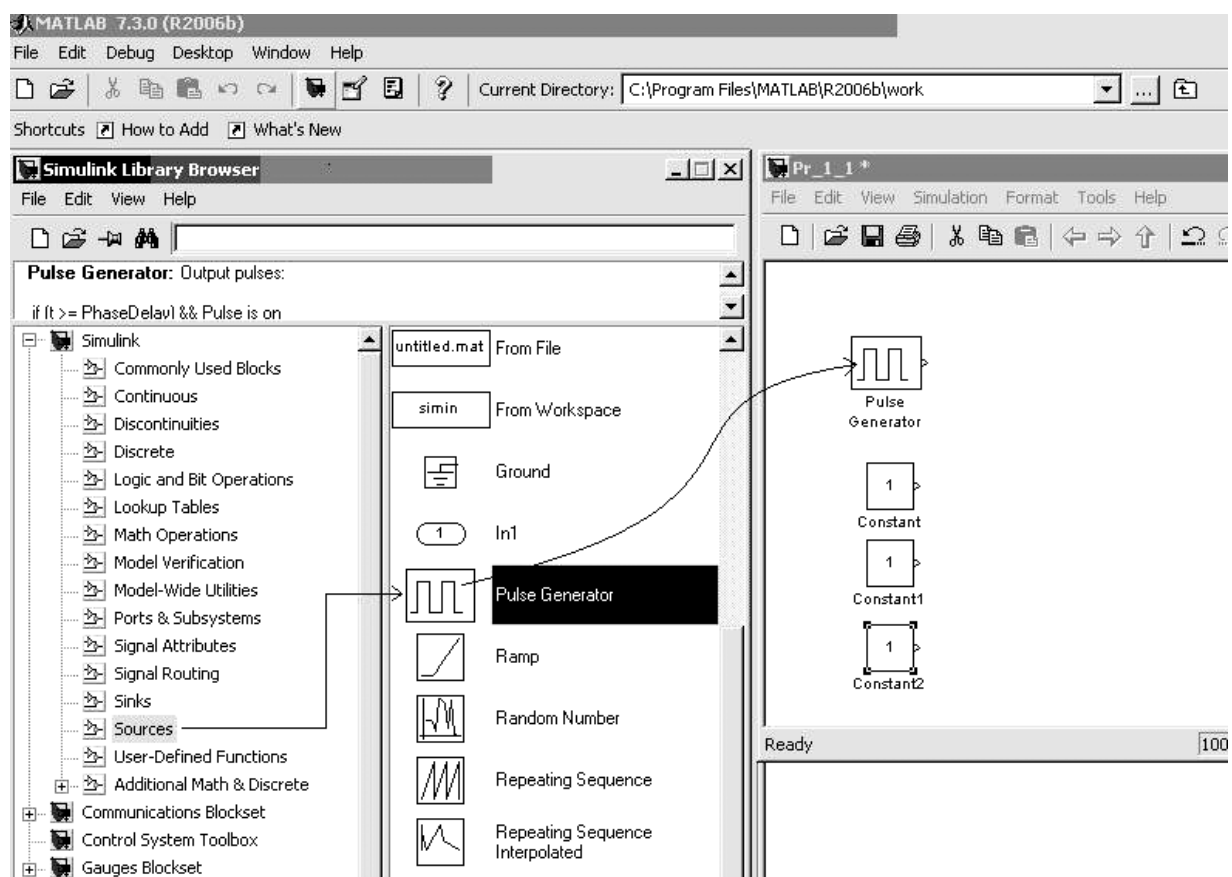


Рис. 1.5

2) Расположим окна Simulink Library browser и Pr_1_1 рядом (рис. 1.5), двойным щелчком левой кнопки мыши выделим в разделе Simulink окна Simulink Library browser раздел Sources, найдем в рас-

крывшемся справа окне модуль Pulse Generator, захватим его левой кнопкой мыши и не отпуская ее перетащим этот модуль в окно Pr_1_1. Таким же образом можно перенести из соответствующих разделов из библиотеки в рабочее окно Pr_1_1 все остальные модули. Альтернативным вариантом переноса является использование команды Edit|Add to current model (Ctrl+I). Выделим блок Constant в упомянутом разделе Sources и введем команду Ctrl+I – этот блок появится в центре окна Pr_1_1. Для построения данной модели нам понадобятся еще два таких же блока (в них будет храниться информация о величинах R и C). Эти блоки создадим копированием блока Constant: нажмем клавишу Ctrl, захватим левой клавишей мыши копируемый блок (рядом с курсором появится знак “+”) и перетащим изображение копии блока на свободное место.

Разместим на поле рабочего окна недостающие компоненты модели (рис. 1.6) :

- а) сумматор Add, два перемножителя Product, блок деления Divide из раздела Maht Operations раздела Simulink библиотеки (Simulink| Maht Operations);
- б) интегратор Integrator (Simulink| Continuous);
- с) осциллограф Scope (Simulink| Sinks).

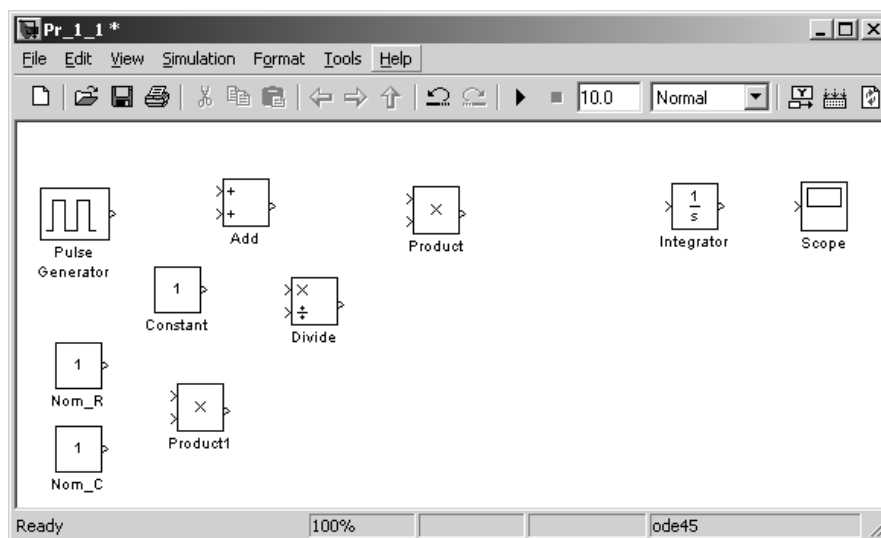


Рис. 1.6

Изменим названия блоков: Constant1 обозначим Nom_R, а Constant2 – Nom_C. Для изменения названия достаточно дважды щелкнуть по нему левой кнопкой мыши и ввести новое название. Сделаем верхний вход сумматора Add инверсным, для чего дважды щелкнув по его изображению, откроем окно установки параметров (рис. 8) и в поле List of signs (список знаков) изменим последовательность знаков “++” на “- +”.

Соединим входы и выходы блоков между собой в соответствии с функциональной схемой модели (рис. 1.3). Для соединения блоков достаточно совместить курсор мыши с входом или выходом одного из соединяемых блоков (форма указателя курсора изменится на “+”), нажать левую кнопку мыши и провести линию к входу или выходу другого блока). Другой вариант выполнения этой операции: выделить блок, выход которого необходимо соединить с каким-либо блоком и нажав клавишу Ctrl щелкнуть левой кнопкой мыши по тому блоку, к которому подключается выделенный блок. Если далее при нажатой клавише Ctrl щелкнуть еще по одному блоку – окажется соединенной цепочка из трех блоков.

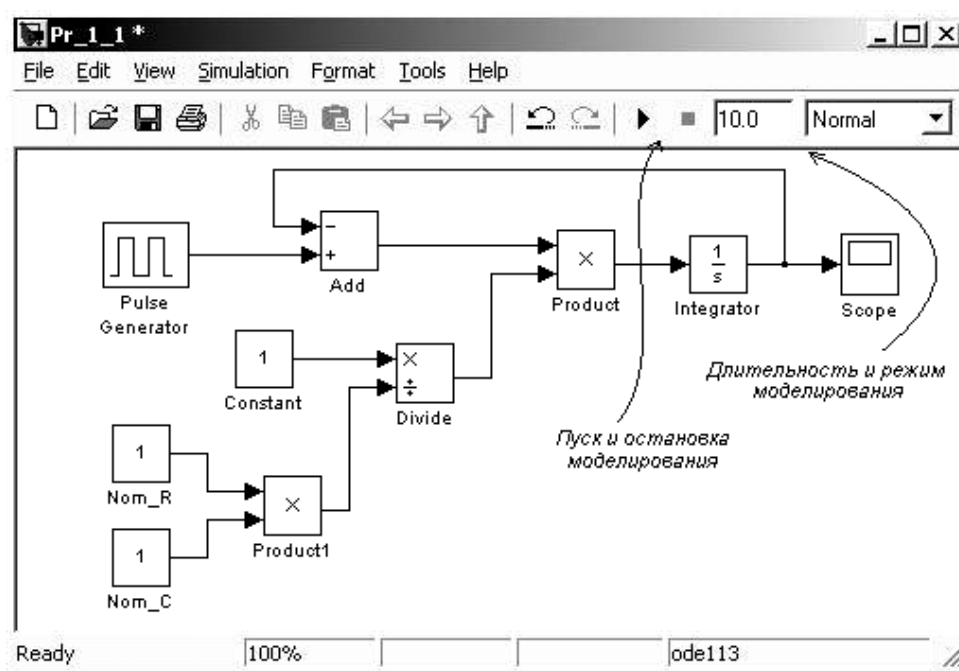


Рис. 1.7

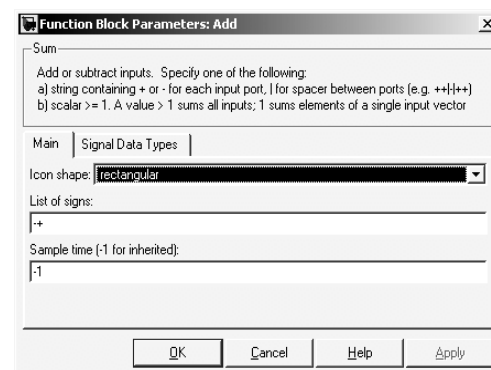
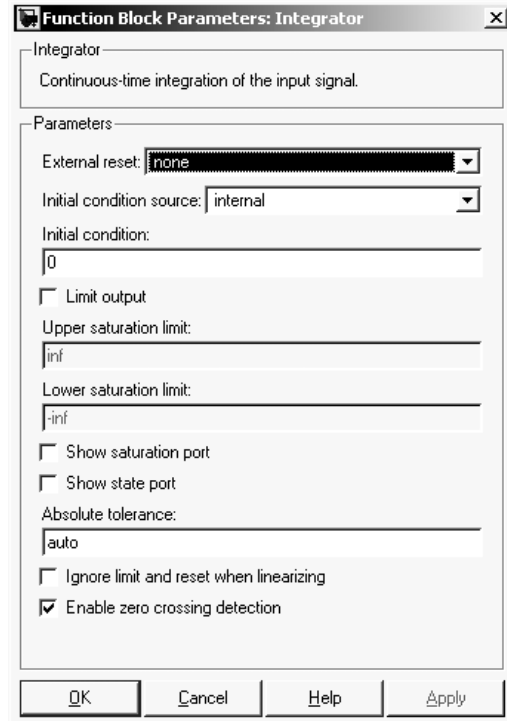


РИС. 1.8

3) Сохраним подготовленную модель (Ctrl+S) и установим необходимые параметры блоков. Установленные по умолчанию параметры блоков можно изменять в окне установки параметров Function Block Parameters (рис. 1.8), которое отрывается либо двойным щелчком мыши по изображению блока в рабочем окне, либо командой Product Parameters, которая активизируется из всплывающего меню. Для вызова всплывающего меню щелкните правой кнопкой мыши по изображению соответствующего блока.

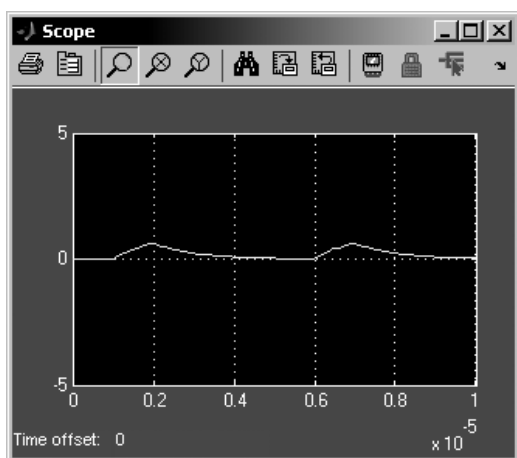
Окно Function Block Parameters содержит информационное поле, в котором после названия блока приведена краткая информация о нем. Под информационным полем располагаются поля для ввода параметров блока и набор кнопок. В некоторых блоках параметры распределены по двум вкладкам Main и Signal Data Types. Набор параметров зависит от вида блока.

Пусть, например моделируемая цепь имеет следующие параметры: $R=1\text{кОм}$, $C=1,5\text{нФ}$. Входной сигнал – последовательность прямоугольных импульсов длительностью 1мкс , периодом повторения 5мкс и амплитудой 1В . В соответствии с этим установим следующие значения параметров модели. Для блоков R и C введем значения Constant value равными $1\text{e}3$ и $1\text{e}-9$ соответственно. Для блока Pulse Generator: Period – $5\text{e}-6$; Pulse Width – 20; Phase Delay – $1\text{e}-6$. Остальные пара-

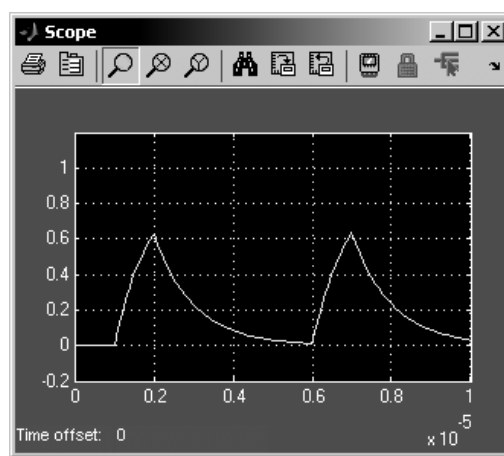
метры оставим без изменения, т.е. такими, какими они установлены по умолчанию.

1.3. Установим интервал моделирования равным $10\text{e-}6$, т.е. 10мкс . Эту операцию можно выполнить двумя способами: ввести $10\text{e-}6$ в поле установки длительности моделирования, расположенное рядом с панелью инструментов рабочего окна (рис. 7) либо в поле Stop time окна конфигурации параметров модели, которое вызывается из рабочего окна командой Simulation|Configuration Parameters (Ctrl+E).

Проведем пробный цикл моделирования, нажав кнопку старта модели на панели инструментов рабочего окна, либо введя команду Simulatin|Start (Ctrl+T). Для наблюдения результатов моделирования двойным щелчком левой кнопки по блоку Scope откройте его экран, на который будут выведена осциллограмма процесса $y(t)$ – напряжения на конденсаторе (рис. 1.9а).



а)



б)

Рис. 1.9

Чтобы изменить масштаб осциллограммы щелкните правой кнопкой мыши по экрану осциллографа, выберите команду Axis properties и в открывшемся диалоговом окне измените значения Ymin и Ymax. Закройте окно нажатием кнопки “ОК” – масштаб осциллограммы на экране изменится на выбранный (рис. 1.9б).

Для верификации модели необходимо испытать ее при таких тестовых сигналах, результаты воздействия которых известны либо

их теории, либо из испытаний реальных систем. В данном случае в качестве критерия верификации можно использовать длительность переходного процесса в RC-цепи при воздействии единичного скачка. Как известно напряжение на конденсаторе в RC-цепи в этом случае изменяется по закону $y(t) = U_0(1 - \exp(-t/RC))$, где U_0 – величина скачка напряжения. Через интервал времени равный $3RC$ напряжение $y(t)$ достигает уровня $0,95 U_0$.

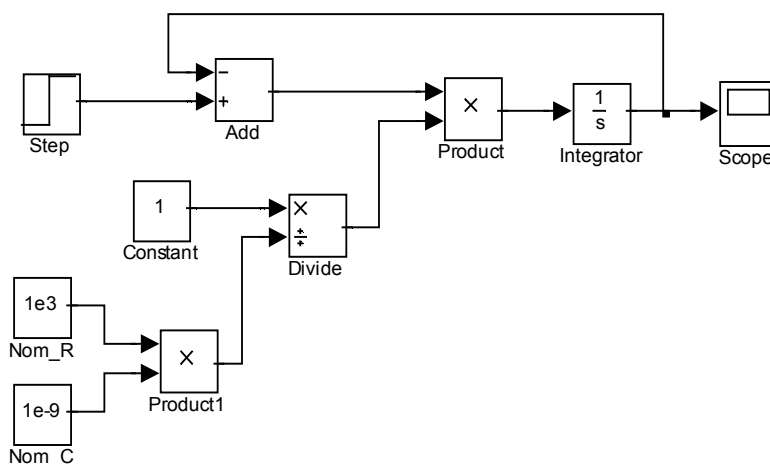


Рис. 1.10

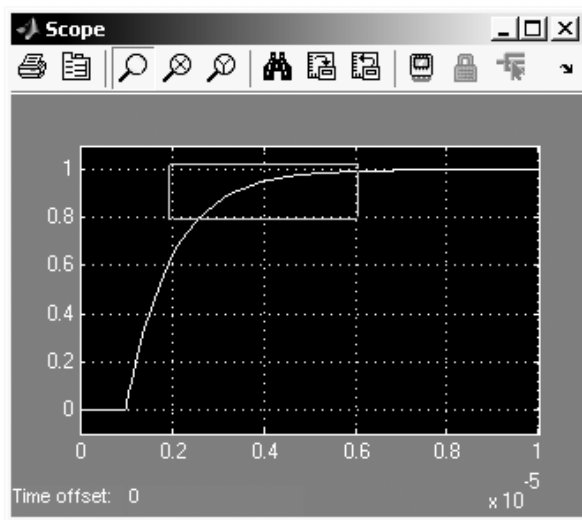
Заменяем в модели (рис.7) генератор импульсов Pulse Generator на генератор единичного скачка – блок Step ((Simulink| Sources) (рис. 1.10). Параметр Step time (начало скачка) установим равным $1e-6$ (1мкс). При $R=1\text{кОм}$, $C=1\text{нФ}$ интервал времени $3RC = 3 \cdot 10^3 \cdot 10^{-9} = 3 \cdot 10^{-6} = 3\text{мкс}$.

На рис. 1.11а показаны результаты моделирования воздействия единичного скачка напряжения на RC-цепь, а на рис. 1.11б – выделенный участок переходной характеристики в диапазоне $(0,8 \div 1)U_0$. Как видно из рис. 1.11б уровня $0,95 y(t)$ достигает при $t_{0,95}=4\text{мкс}$. Поскольку скачок начинается в момент времени $t_0=4\text{мкс}$, то длительность переходного процесса составит 3мкс, что соответствует ожидаемым результатам.

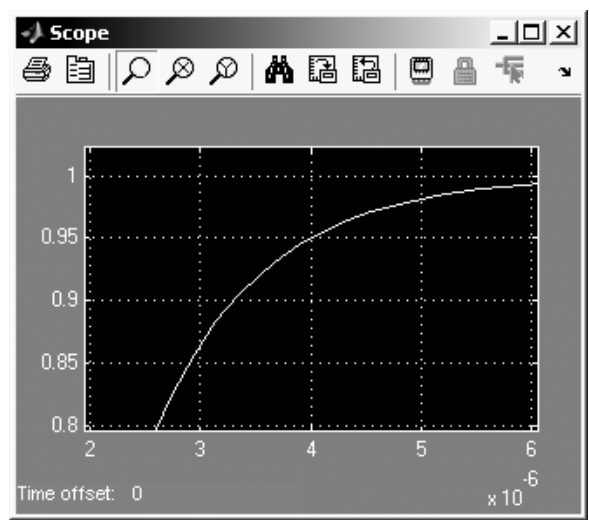
Примечание. Для просмотра в более крупном масштабе какого-либо участка осциллограммы достаточно выделить его, нажав левую кнопку мыши. Чтобы вернуться к исходному изображению, щелкните по “экрану” осцилло-

графа правой кнопкой мыши и во всплывающем меню выберите одну из команд *Autoscale* или *Axis properties*.

Изменим один из параметров R или C. Возьмем, например, $C=2$ нФ. Результаты моделирования приведены на рис. 1.12. Как показывает их анализ, длительность переходного процесса до уровня $0,95U_0$ составляет 6 мкс, что соответствует $3RC = 3 \cdot 10^3 \cdot 2 \cdot 10^{-9} = 6 \cdot 10^{-6}$ с.

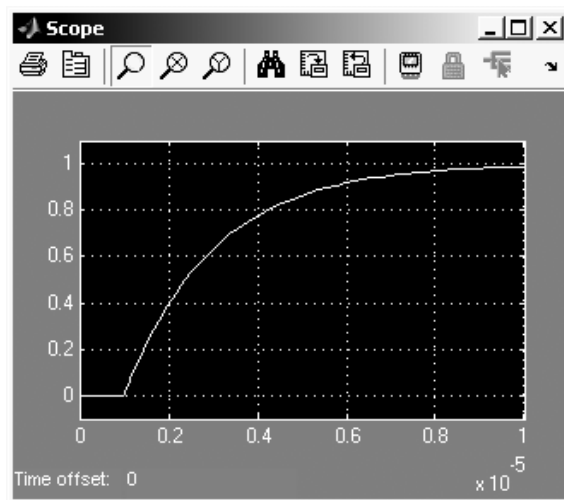


а)

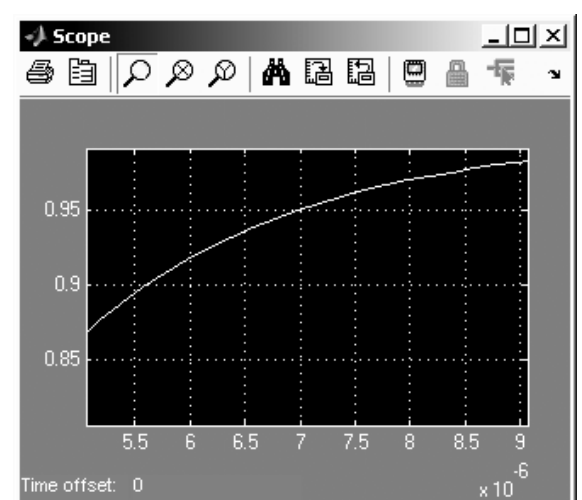


б)

Рис. 1.11



а)



б)

Рис. 1.12

Таким образом, испытания модели показали, что полученные на ней результаты не противоречат данным, принятым в качестве эталонных для верификации. Это позволяет сделать вывод о том, что

модель адекватно отображает происходящие в исследуемой системе процессы.

1.5. Покажем, как можно передавать данные из MatLab в Simulink-модель. Для этого удалим из модели (рис. 10) блоки Nom_R, Nom_C и Product1 и заменим их блоком Constant1 (рис. 1.13), причем в поле Constant value при установке параметров блока вместо численного значения введем имя переменной, которой обозначим произведение $R \cdot C$. Выберем в качестве такого имени, например *RC*.

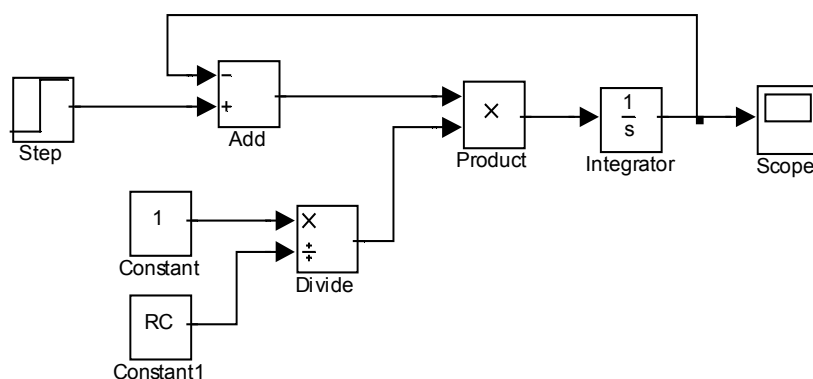


Рис. 1.13

Перейдем в командное окно, введем в него команду “ $RC=1e-6;$ ” (это соответствует $R=1\text{кОм}$, $C=1\text{ нФ}$), вернемся в рабочее окно Simulink и выполним моделирование, введя команду Simulation|Start (Ctrl+T). Сравним полученные результаты с результатами, показанными на рис. 11 и убедимся в их идентичности. Введя в командное окно MatLab “ $RC=2e-6;$ ” и вновь выполнив моделирование в Simulink, получим результаты, показанные на рис. 1.12. Необходимо иметь ввиду, что переменная *RC* хранится в рабочей области (Workspace) MatLab, поэтому она не сохраняется при выходе из него. При повторном запуске MatLab необходимо либо вновь задать значения таких переменных, либо загрузить ранее сохраненное состояние рабочей области.

Для сохранения рабочего пространства перед выходом из MatLab необходимо ввести в его командном окне одну из следующих команд:

Save ('<имя файла>') - для сохранения в файле всей рабочей области MatLab;

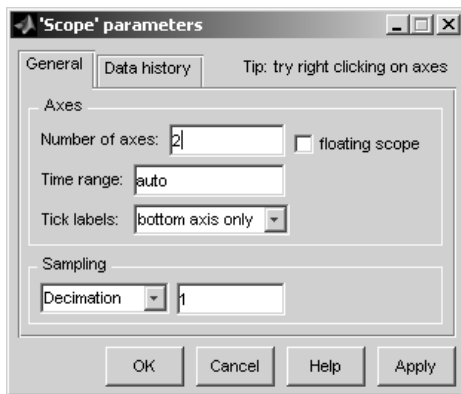
Save ('<имя файла>', ' <var1>', ' <var2>', ...) - для сохранения в файле переменных с именами *var1*, *var2* и т.д.

Чтобы загрузить ранее сохраненное рабочее пространство или переменные, используются соответственно команды Load ('<имя файла>') или Load ('<имя файла>', ' <var1>', ' <var2>', ...).

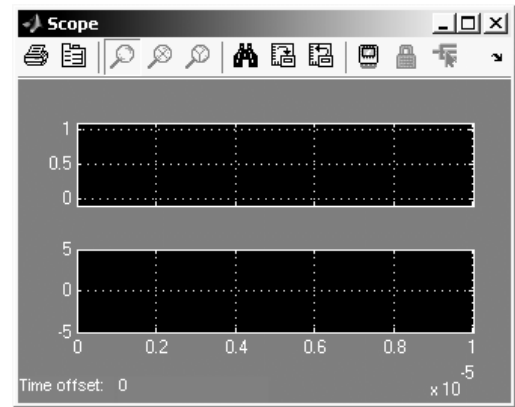
1.6. Для более наглядного отображения результатов моделирования на “экран” осциллографа можно выводить одновременно несколько процессов. Модернизируем модель, показанную на рис. 1.13 следующим образом.

Заменим генератор единичного скачка Step генератором импульсов Pulse Generator, установив в нем указанные ранее параметры: Period – 5e-6; Pulse Width – 20; Phase Delay – 1e-6. Далее двойным щелчком мыши по блоку Scope в рабочем окне откроем его переднюю панель и щелкнем по кнопке Parameters (вторая слева) на панели инструментов. В результате этих действий откроется окно ‘Scope’ parameters (рис. 1.14а), в котором поле Number of axes предназначается для указания количества одновременно наблюдаемых сигналов. Введем в это поле цифру 2 и нажмем кнопку ОК – изменится вид “экрана” осциллографа (рис. 1.14б), а на изображении блока Scope в рабочем окне появится еще один вход. Соединим его с выходом блока Pulse Generator (рис. 1.15а) и выполним моделирование (рис. 1.15б).

На поле модели можно наносить поясняющие надписи и комментарии (рис. 1.4). Чтобы сделать это достаточно дважды щелкнуть левой кнопкой мыши по тому месту рабочего окна модели, где будет вводиться текст, и ввести требуемые пояснения. К сожалению Simulink отказывается сохранять файлы моделей, если такие файлы содержат символы кириллицы даже в комментариях.

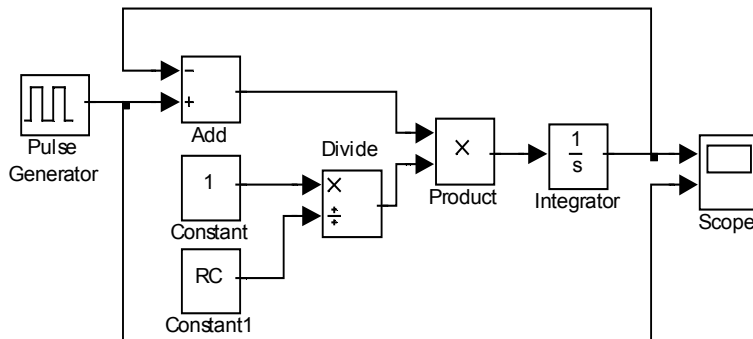


a)

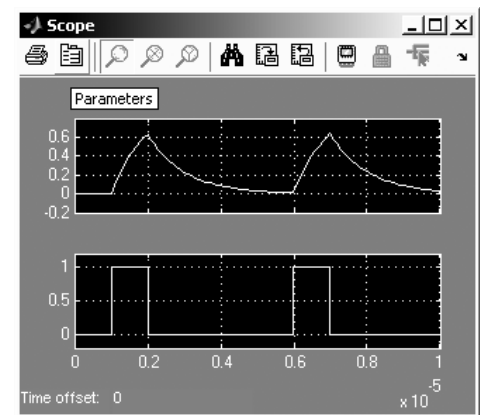


б)

Рис.1.14



a)



б)

Рис. 1.15

1.7. В заключение рассмотрим некоторые полезные приемы редактирования модели. Прежде всего, отметим, что Simulink поддерживает все стандартные для Windows-приложений приемы редактирования текстовых и графических файлов: перемещение, растяжение и сжатие объектов, их копирование через буфер обмена и т. д. При перемещении блока ранее введенные связи с другими блоками не разрушаются.

Блоки можно вращать (Ctrl+R), поворачивать вокруг оси (Ctrl+I), изменять их наименования (с ограничениями на применение символов кириллицы). Для соединения блоков между собой достаточно совместить курсор с входом или выходом блока и, нажав левую

кнопку мыши, переместить курсор на вход или выход другого блока. Чтобы сделать отвод от соединительной линии, необходимо переместить указатель курсора в точку, от которой предполагается сделать отвод, нажать клавишу Ctrl и провести требуемую линию. Отрезки соединительных линий можно перемещать, захватив их левой кнопкой мыши. При нажатой клавише Shift можно проводить наклонные соединительные линии или деформировать ранее введенные прямые.

Для удаления соединительной линии, ее необходимо выделить и нажать клавишу Delete либо после выделения выбрать эту команду из всплывающего меню, которое активизируется щелчком правой кнопки мыши. Если соединительная линия достаточно длинная, то в ее разрыв можно вставить новый блок, не удаляя этой линии. Для этого достаточно наложить этот блок на линию, совместив его входы и выходы с ней.

1.8. Рассмотрим подробнее приемы работы с блоком Score – осциллографом. Блок показывает временные диаграммы сигналов, подключенных к его входу, и предназначен для наблюдения за изменениями сигналов в процессе моделирования. Блок может отображать многокомпонентный (векторный) сигнал. В этом случае каждая компонента сигнала отображается своим цветом (см. файл-пример Score_.mdl): первая – желтым, вторая сиреневым, третья – голубым и т.д.

Блок позволяет масштабировать наблюдаемые процессы вручную, устанавливая автоматически масштабы по обеим осям, изменять число входов, связывать наблюдаемые процессы с сигналами модели не графически, а по имени (режим Floating Scope) и т.д. Настройка окна осциллографа выполняется с помощью панели инструментов (рис. 1.16), на которой расположено 12 кнопок:



Рис. 1.16

1. Print – печать содержимого окна осциллографа;
2. Parameters – доступ к окну установки параметров;
3. Zoom – увеличение масштаба по обеим осям;
4. Zoom X- axes – увеличение масштаба по горизонтальной оси;
5. Zoom Y- axes – 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, когда наблюдаемые процессы связываются с сигналами модели не графически, а по именам;
10. Lock/Unlock axes selection (доступна в режиме Floating scope) – закрепить/разорвать связь между текущей координатной системой блока и наблюдаемым сигналом
11. Signal selection (доступна в режиме Floating scope) – выбор сигналов для отображения;
12. Dock Time Scope – разместить окно блока на компоновочную панель графиков MatLab.

Изменить масштаб отображаемых графиков можно несколькими способами:

1) нажать одну из кнопок 1, 2 или 3 (рис. 1.16) и левой кнопкой мыши щелкнуть в нужном месте графика – масштаб увеличится в 2,5 раза;

2) нажать одну из кнопок 1, 2 или 3 (рис. 1.16) и, нажав левую кнопку мыши, с помощью динамической рамки или отрезка указать область графика для увеличенного изображения;

3) щелкнуть правой кнопкой мыши по окну вывода графика, во всплывающем меню выбрать команду Axes properties... и в открыв-

шемся окне ‘Scope’ properties в полях **Y-min** и **Y-max** указать предельные значения вертикальной оси графика. В поле **Title** можно указать заголовок графика (латинским шрифтом).

Параметры блока устанавливаются в окне ‘Scope’ parameters (рис. 1.17), которое открывается нажатием кнопки 2 (рис. 1.16)

В этом окне на вкладке General задаются следующие параметры: Number of axes – число входов осциллографа; Time range – временной интервал, для которого отображаются графики; Tick labels – скрыть/показать метки и подписи к осям; Sampling – режим вывода точек на экран: в режиме Decimation указывается кратность вывода точек на экран (например, если Decimation = 5, то будет выводиться

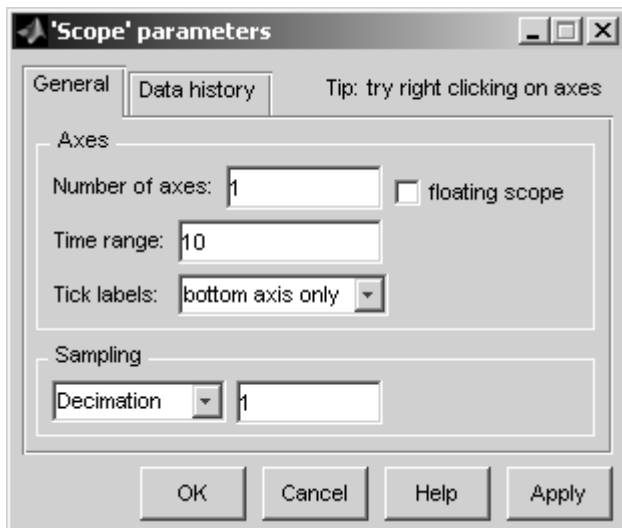


Рис. 1.17

каждая 5-я точка); в режиме Sample time точки выводятся через интервал модельного времени равный Sample time. Флаг Floating scope можно использовать для перевода осциллографа в “свободный режим”.

На вкладке Data History задается максимальное число отображаемых расчетных точек графика Limit data points to last.

При превышении этого числа начальная часть графика обрезается. Сбросив флаг параметра Limit data points to last, можно снять это ограничение и вывести на экран весь график (соответственно увеличивается расход ресурса памяти под модель).

Отображаемые данные можно передать в рабочее пространство MatLab установкой флага Save data to workspace, задав имя переменной в поле Variable name и выбрав формат из списка Format.

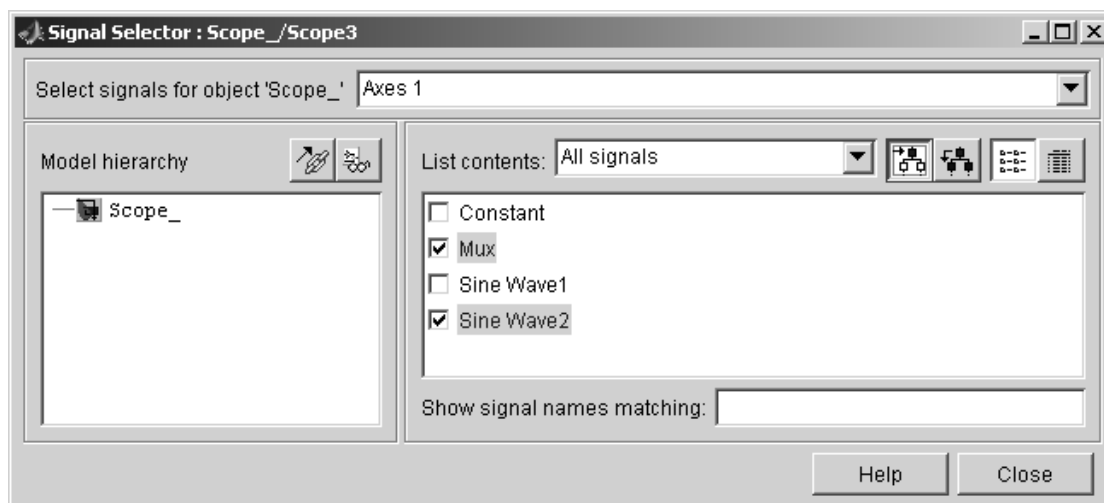


Рис. 1.18

Для наблюдения сигналов в сложных моделях удобно использовать осциллограф в режиме Floating scope. В этом режиме осциллограф не имеет входов, а выбор сигналов осуществляется с помощью инструмента 11 – Signal selection (рис. 1.16). С его помощью открывается окно Signal selector (рис. 1.18) и в нем отмечаются флажками имена блоков, выходные сигналы которых должны отображаться осциллографом. В окне List Contents могут отображаться как сигналы модели только текущего уровня иерархии (когда нажата кнопка Current System), так и сигналы подсистем более низких уровней (при нажатой кнопке Current System and below). В списке Select signals for object указывается вход осциллографа, на который подаются выбранные в окне List Contents сигналы.

1.8. Мощным средством Simulink является возможность выделения отдельных фрагментов модели в подсистемы, которые оформляются в виде блоков. Это позволяет облегчить зрительное восприятие модели (в идеале она должна размещаться на экране монитора), дает возможность отлаживать модель по фрагментам, создавать собственные библиотеки, позволяет синхронизировать работу параллельно работающих подсистем. Подсистема, в свою очередь может разбиваться на подсистемы более низких уровней иерархии, что позволяет структурировать модель и распределить выполняемые операции по

уровням иерархии, а также облегчает модернизацию модели и ее адаптацию к решению задач, выходящих за пределы первоначально определенных рамок.

Связь подсистемы с моделью или подсистемой верхнего уровня иерархии осуществляется через входные (Inport) и выходные (Outport) порты, имена которых могут изменяться со стандартных (In1, In2,..., Out1, Out2,... и т.д.) на те, которые нужны пользователю. Подсистемы могут быть виртуальными (Subsystem) и монолитными (Atomic Subsystem). Монолитная подсистема с точки зрения Simulink неделимый блок и расчет в таком блоке выполняется от начала до конца без переключения на другие блоки. В виртуальной подсистеме при расчете игнорируются границы между ней и моделью, и расчеты ее блоков могут перемежаться с расчетами блоков других подсистем и блоков модели. Изображение монолитной подсистемы имеет более толстую рамку в сравнении с изображением виртуальной подсистемы.

Подсистемы разделяют также на управляемые и неуправляемые. Управляемые имеют управляющие входы, на которые подаются сигналы, активизирующие их работу. Такие системы всегда являются монолитными. В активном состоянии управляемая подсистема выполняет вычисления, в пассивном - не выполняет, а ее выходные сигналы определяются ее настройками управляющего порта..

Для создания подсистем можно применить один из двух следующих способов:

- 1) скопировать из библиотеки Simulink (раздел Ports& Subsystems) в модель шаблон нужной подсистемы и наполнить его конкретными блоками и связями в соответствии с моделью данной подсистемы;

- 2) выделить с помощью мыши нужный фрагмент модели и ввести команду Edit| Create Subsystem – выделенный фрагмент модели будет

ды щелчком по изображению подсистемы (откроется окно с ее моделью) и переименуем в ней порты In1 на Pulse, а In2 на RC (рис. 1.19б). Далее введем в блок Constant1 значение константы $RC=2e-6$ и выполним моделирование системы (рис. 1.19в).

Превратим созданную подсистему в управляемую. Для этого скопируем из раздела Ports& Subsystems библиотеки Simulink компонент Enable. При этом в подсистему добавится управляющий порт, по которому на ее вход подается управляющий сигнал, положительное значение которого разрешает работу подсистемы (рис. 1.20а, б).

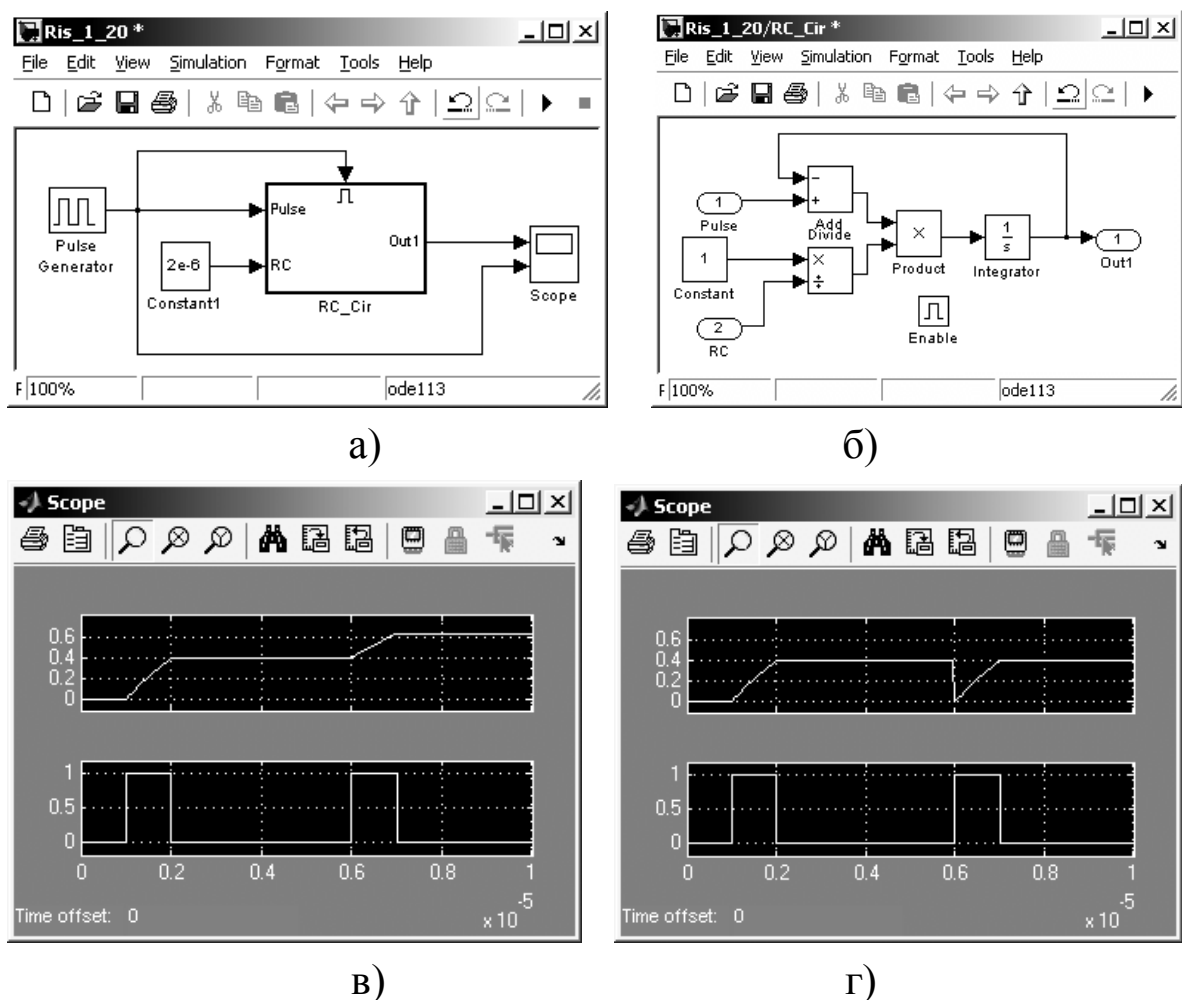


Рис. 1.20

Такие управляемые подсистемы получили название Е-подсистем. Состояние подсистемы в начале следующего цикла расчета определяется значением параметра **States when enabling** компонента Enable.

Если это значение held – в паузах между циклами расчета на выходе сохраняется последнее значение выходного сигнала в предыдущем цикле (рис. 1.20г). Когда выбрано значение reset выходной сигнал перед каждым циклом обнуляется (рис. 1.20д).

2. МЕТОДЫ ПОСТРОЕНИЯ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

2.1. Основные понятия математического моделирования

Моделирование можно рассматривать как замещение исследуемого объекта его условным образом, описанием или другим объектом, именуемым *моделью* и обеспечивающим адекватное с оригиналом поведение в рамках некоторых допущений и приемлемых погрешностей [2, 4]. Целью моделирования является изучение некоторых свойств исследуемого объекта путем исследования его модели, а не самого объекта.

Математическая модель – это совокупность математических уравнений и логических соотношений, описывающих исследуемый объект в рамках решаемой задачи на уровне, достаточно близком к реальному поведению объекта при его натурных испытаниях.

Математическое моделирование это процедура, которая обеспечивает исследование протекающих в моделируемом объекте процессов на основе его математической модели. При этом используются фундаментальные положения и законы математики, описывающие моделируемые явления, системы или устройства на некотором уровне их идеализации.

Критерием корректности модели обычно считают достаточно малую погрешность (чаще всего среднеквадратическую) результатов моделирования.

Одним из наиболее ответственных этапов математического моделирования является построение математической модели, т. е. системы уравнений, описывающей поведение моделируемого объекта.

Методы построения таких систем уравнений зависят от решаемой при исследовании задачи. Например, сложные электрические цепи постоянного тока легко описываются системами линейных уравнений с вещественными коэффициентами, составленными по зако-

нам Кирхгофа. Для исследования частотных характеристик цепей переменного тока составляют такие уравнения с комплексными коэффициентами. А для моделирования динамических систем и устройств во временной области составляют системы дифференциальных уравнений, чаще всего нелинейных. Если внутренняя структура исследуемой системы не известна или достаточно сложна, может оказаться целесообразным рассматривать ее как “черный ящик” и представлять при моделировании некоей эквивалентной системой. Под эквивалентностью систем в данном случае понимается близость реакций двух систем на некоторые тестовые воздействия. В связи с этим при построении моделей динамических систем очень важно уметь представлять моделируемую систему разными способами и уметь переходить от одного представления к другому.

Система уравнений, описывающая моделируемую динамическую систему, может решаться аналитически либо численно. В соответствии с этим иногда говорят об *аналитическом* и *численном* моделировании [2, 4]. Известен обширный класс систем, состояние которых может изменяться скачкообразно под воздействием как внешних, так и внутренних факторов. Это автоматические системы, состоящие из нескольких подсистем, каждая из которых предназначена для функционирования в определенной ситуации. Для исследования таких систем применяют так называемое *ситуационное* моделирование [2, 7].

Результатом *аналитического моделирования* являются аналитические выражения, описывающие реакцию системы на те или иные воздействия.

Однако даже в простейших случаях потребуются навыки и существенные умственные усилия для решения дифференциального уравнения, описывающего работу системы. Другим вариантом решения задачи аналитического моделирования является применение для решения систем компьютерной математики с символьными вычислениями (например, пакетов символьной математики Mathematics, Maple или Symbolic Math Toolbox - пакета специального расширения MatLab), позволяющих в той или иной мере автоматизировать процедуру нахож-

дения аналитического решения. Однако возможности аналитического моделирования ограничиваются исследованиями линейных динамических систем второго – третьего порядка в силу трудностей осмысливания полученных аналитических выражений.

При *численном моделировании* дифференциальные уравнения, описывающие исследуемую динамическую систему, решаются численными методами. Численное моделирование предполагает решение рассматриваемых уравнений достаточно апробированными и хорошо известными численными методами. Результатом численного решения являются массивы чисел или вектора, описывающие реакцию системы на внешние воздействия. Эти результаты можно представить в графическом виде, интерпретировать в виде временных диаграмм, частотных характеристик и т.д. Уступая в общности аналитическому, численное моделирование привлекательно тем, что оно применимо к широкому классу систем и не требует от исследователя высокой математической культуры.

MatLab предоставляет исследователю обширные возможности для численного моделирования линейных и нелинейных систем и устройств, которые описываются системами уравнений высоких порядков. Для решения систем уравнений можно использовать ряд различных методов, в том числе на основе рекуррентных и итерационных алгоритмов. Это дает возможность на этапах верификации модели сравнить результаты, полученные разными методами и выбрать тот из методов, который в данных конкретных условиях в наибольшей степени удовлетворяет исследователя. Уравнения состояния реальных систем и устройств часто содержат множество нулевых коэффициентов, что порождает разреженные матрицы и массивы. Их аппарат также прекрасно представлен в базовой системе MatLab. Simulink, являясь надстройкой над MatLab, в полной мере использует эти методы.

Ситуационное моделирование реализовано в Simulink в одном из его расширений State Flow. В нем реализована концепция событийного моделирования, когда система скачком переходит из одного состояния в другое под воздействием некоторых событий, происходящих как внутри

исследуемой системы, так и вне ее. Классическим примером таких моделей являются конечные автоматы Мура и Мили [3, 7], широко используемые для моделирования и проектирования цифровых систем управления. При событийном моделировании в Simulink модель динамической системы представляется в виде так называемой SF-диаграммы – построенной по определенным правилам графической диаграммы состояний системы и переходов между ними. Применение SF-диаграмм особенно удобно для описания цифровых динамических систем, поскольку этот метод позволяет абстрагироваться от внутренней структуры системы, ограничившись описанием ее поведения.

Систему называют *линейной*, если для нее выполняется принцип суперпозиции: реакция системы на сумму сигналов равна сумме реакций на сигналы, действующих по отдельности. Системы, для которых этот принцип не выполняется, называют *нелинейными*.

Систему называют *стационарной* или *системой с постоянными параметрами*, если произвольная задержка подаваемого на вход сигнала приводит лишь к такой же задержке выходного сигнала без изменения его формы. В противном случае система называется *нестационарной, параметрической* или *системой с переменными параметрами*.

Систему называют непрерывной, если состояние системы может изменяться в любые моменты времени. Систему называют дискретной, если ее моменты времени, в которые ее состояние может изменяться, образуют некоторое счетное множество (конечное или бесконечное). Если множество состояний, в которых может находиться система счетное, то такую систему называют цифровой.

2.2. Методы описания непрерывных линейных систем

Непрерывная стационарная линейная система может быть описана одним из следующих способов [6]:

- 1) с помощью импульсной характеристики;
- 2) с помощью переходной характеристики;
- 3) с помощью комплексного коэффициента передачи;

4) с помощью операторного коэффициента передачи (в полиномиальной или дробно-рациональной форме, набором нулей и полюсов или набором полюсов и вычетов);

5) с помощью дифференциального уравнения;

6) с помощью вектора состояния системы в некотором пространстве состояний (state space).

Поскольку вышеперечисленные способы описывают один и тот же физический объект, то они являются эквивалентными в том смысле, что, используя любое из описаний, исследователь получит, в конечном счете, один и тот же результат. Однако эти способы приводят к разным моделирующим уравнениям и далеко не эквивалентны с точки зрения удобства проведения моделирования при решении разных задач. Поэтому важно уметь переходить от одного способа к другому. Рассмотрим варианты решения этой задачи.

Импульсной характеристикой $h(t)$ системы называют ее реакцию на воздействие тестового сигнала в виде δ – функции (на практике – очень короткого импульса с формой близкой к прямоугольной). Как известно [6] выходной сигнал $y(t)$ линейной системы связан со входным сигналом $x(t)$ через интеграл Дюамеля

$$y(t) = \int_{-\infty}^t x(t - \tau) h(\tau) d\tau \quad (2.1)$$

Переходной характеристикой $g(t)$ системы называют ее реакцию на воздействие тестового сигнала в виде функции единичного скачка $u(t)$.

$$u(t) = \begin{cases} 0 & \text{при } t < 0, \\ 1 & \text{при } t \geq 0. \end{cases}$$

$g(t)$ и $h(t)$ следующим образом связаны между собой:

$$h(t) = \frac{d g(t)}{dt}, \quad g(t) = \int_{-\infty}^t h(\tau) d(\tau). \quad (2.2)$$

Примечание. В отечественной научно-технической литературе обычно применяют другие обозначения: $g(t)$ - для импульсной характеристики $h(t)$. – для

переходной. Отступление от этого правила в настоящем пособии объясняется тем, что такие обозначения используются в документации по MatLab.

Комплексный коэффициент передачи $\dot{K}(j\omega)$ и импульсная характеристика $h(t)$ связаны между собой парой преобразований Фурье:

$$\begin{aligned}\dot{K}(j\omega) &= \int_{-\infty}^{\infty} h(t) \exp(-j\omega t) dt \\ h(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{K}(j\omega) \exp(j\omega t) d\omega\end{aligned}\quad (2.3)$$

Операторный коэффициент передачи $K(s)$ (другое название – передаточная функция) импульсная характеристика $h(t)$ связаны между собой парой преобразований Лапласа:

$$\begin{aligned}K(s) &= \int_0^{\infty} h(t) \exp(-st) dt \\ h(t) &= \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} K(s) \exp(st) ds ,\end{aligned}\quad (2.4)$$

где $s = \sigma + j\omega$ – лапласовская переменная.

Для физически реализуемых линейных динамических систем с сосредоточенными параметрами $K(s)$ имеет вид дробно-рациональной функции

$$K(s) = \frac{B(s)}{A(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} ; \quad n > m . \quad (2.5)$$

Выражение (2.5) часто представляют в одной из следующих форм:

$$K(s) = k \frac{(s - z_m)(s - z_{m-1})(s - z_{m-2}) \dots (s - z_1)}{(s - p_n)(s - p_{n-1})(s - p_{n-2}) \dots (s - p_1)} , \quad (2.6)$$

или

$$K(s) = \frac{r_n}{(s - p_n)} + \frac{r_{n-1}}{(s - p_{n-1})} + \frac{r_{n-2}}{(s - p_{n-2})} + \dots + \frac{r_1}{(s - p_1)} + C . \quad (2.7)$$

В формулах (2.6), (2.7) используются следующие обозначения:

z_i и p_i - соответственно корни числителя $B(s)$ и знаменателя $A(s)$ операторного коэффициента передачи (2.5), которые принято называть нулями (zero) и полюсами (pole) передаточной функции;

r_i – вычет функции $K(s)$ относительно полюса p_i ;

k и C – константы, первая из которых имеет смысл коэффициента усиления системы, а вторая отлична от нуля только в случае равенства степеней полиномов $B(s)$ и $A(s)$.

Примечание. Для линейных электрических схем операторный коэффициент передачи (2.5) можно получить непосредственно по электрической схеме устройства (см. пример 2.1).

Нули и полюса передаточной функции $K(s)$ определяют форму частотной характеристики исследуемой системы: на частотах ω_i , соответствующих нулям z_i передаточной функции ее амплитудно-частотная характеристика $|\dot{K}(j\omega)|$ имеет минимумы, а на частотах соответствующих полюсам p_i – максимумы. Для физически реализуемых систем полюсы p_i могут быть вещественными или комплексно сопряженными. Комплексно-сопряженной паре полюсов \dot{p}_i и p_i^* соответствует пара комплексно сопряженных вычетов \dot{r}_i и r_i^* .

Представление передаточной функции в форме (2.7) позволяет вычислить импульсную характеристику системы, которая в случае отсутствия у полинома $A(s)$ кратных корней определяется формулой:

$$h(t) = \sum_{i=1}^n r_i \exp(p_i t). \quad (2.8)$$

В случае кратных полюсов выражение для вычисления $h(t)$ несколько усложняется [4, 6]. Как следует из формулы (2.8), импульсная характеристика рассматриваемых систем представляет сумму экспоненциальных функций. Каждой комплексно-сопряженной паре полюсов соответствует гармоническое колебание с изменяющейся амплитудой:

$$\begin{aligned} \dot{r}_i \exp(\dot{p}_i t) + r_i^* \exp(p_i^* t) &= (\alpha_i + j\beta_i) \exp(\sigma_i + j\omega_i t) + \\ &+ (\alpha_i - j\beta_i) \exp(\sigma_i - j\omega_i t) = 2\sqrt{\alpha_i^2 + \beta_i^2} \exp(\sigma_i t) \cos(\omega_i t + \varphi_i). \end{aligned} \quad (2.9)$$

Начальная фаза φ_i гармонического колебания в формуле (2.9) определяется следующим образом:

$$\varphi_i = \arctg \left(\frac{\beta_i}{\alpha_i} + \psi_i \right).$$

Величина ψ_i в последней формуле принимает одно из значений $(0, \pi/2, \pi, 3\pi/2)$ в зависимости от квадранта, в котором находится вычет r_i .

Как следует из формулы (2.9), необходимым условием устойчивости линейной системы является нахождение всех полюсов ее передаточной функции в левой части комплексной полуплоскости (т.е. их вещественные части должны быть отрицательными). В этом случае все множители $\exp(\sigma_i t)$ и $\exp(p_i t)$ в формулах (2.8) и (2.9) будут убывать с ростом t и любое возмущение системы внешней силой затухать после того, как это воздействие закончилось.

Таким образом, любая физически реализуемая линейная динамическая система с сосредоточенными параметрами в полной мере определяется либо множествами (в терминологии MatLab векторами) коэффициентов $\mathbf{a} = \{a_n, a_{n-1}, \dots, a_1, a_0\}$ и $\mathbf{b} = \{b_m, b_{m-1}, \dots, b_1, b_0\}$, либо множествами нулей и полюсов $\mathbf{z} = \{z_m, z_{m-1}, \dots, z_1\}$ $\mathbf{p} = \{p_n, p_{n-1}, \dots, p_1\}$ и константой k , либо множествами вычетов и полюсов $\mathbf{r} = \{r_n, r_{n-1}, \dots, r_1\}$ $\mathbf{p} = \{p_n, p_{n-1}, \dots, p_1\}$ и константой C_0 .

От представления системы в форме операторного коэффициента передачи $K(s)$ нетрудно перейти к представлению в форме комплексного коэффициента передачи (2.4), если заменить в выражениях (2.5 - 1.7) переменную s на $j\omega$.

Из представления (2.5) можно получить модель системы в форме дифференциального уравнения:

$$\begin{aligned} a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + a_{n-2} \frac{d^{n-2} y}{dt^{n-2}} + \dots + a_1 \frac{dy}{dt} + a_0 y(t) = \\ b_m \frac{d^m x}{dt^m} + b_{m-1} \frac{d^{m-1} x}{dt^{m-1}} + b_{m-2} \frac{d^{m-2} x}{dt^{m-2}} + \dots + b_1 \frac{dx}{dt} + b_0 x(t), \end{aligned} \quad (2.10)$$

где $x(t)$ и $y(t)$ – соответственно входной и выходной сигналы.

Коэффициенты a_i и b_i в формуле (2.10) суть коэффициенты полиномов $A(s)$ и $B(s)$ в формуле (2.5). Безусловно, возможен и обратный переход от выражения (2.10) к выражению (2.5).

Удобным способом описания линейной системы при матричных вычислениях является ее представление в пространстве состояний (state space). Если состояние системы описывается некоторым вектором состояния $\|s(t)\|$, то собственные колебания системы и ее реакция $y(t)$ на входной сигнал $x(t)$ связаны между собой следующим образом:

$$\begin{aligned} \|s'(t)\| &= \|A\| \|s(t)\| + \|B\| x(t) \\ y(t) &= \|C\| \|s(t)\| + D x(t) \end{aligned} \quad (2.11)$$

или в развернутом виде

$$\begin{aligned} \begin{bmatrix} \frac{ds_1}{dt} \\ \frac{ds_2}{dt} \\ \vdots \\ \frac{ds_n}{dt} \end{bmatrix} &= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_n(t) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} x(t) \\ y(t) &= [c_1 \quad c_2 \quad \cdots \quad c_n] \begin{bmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_n(t) \end{bmatrix} + D x(t) \end{aligned} \quad (2.12)$$

В соотношениях (2.11, 1,12) описанием системы является набор параметров: матрицы $\|A\|$, $\|B\|$, $\|C\|$ и скаляр D . Значения этих параметров зависят от структуры и параметров исследуемой системы выбора вектора состояния. В качестве вектора состояний можно выбрать, например, некоторое множество сигналов в системе, либо выходной сигнал и его производные и т.д.

От представления цепи в пространстве состояний можно легко перейти к операторному коэффициенту передачи [4, 6]

$$K(s) = D - \|C\|(\|A\| - s\|I\|)^{-1}\|B\| \quad (2.13)$$

здесь $\|I\|$ — единичная матрица размером $n \times n$.

Реализованный в MatLab алгоритм перехода от $K(s)$ (2.5) или дифференциального уравнения (2.10) к уравнениям (2.11) описан в работе [4].

В качестве основы для построения вектора состояний выбран сигнал $y_0(t)$, являющийся решением дифференциального уравнения

$$a_n \frac{d^n y_0}{dt^n} + a_{n-1} \frac{d^{n-1} y_0}{dt^{n-1}} + a_{n-2} \frac{d^{n-2} y_0}{dt^{n-2}} + \dots + a_1 \frac{dy_0}{dt} + a_0 y_0(t) = b_0 x(t). \quad (2.14)$$

Данное уравнение является частным случаем уравнения (2.10), когда $b_1 = b_2 = b_3 = \dots = b_m = 0$. Выходной сигнал $y(t)$ системы, определяемой выражениями (2.5) или (2.10), выражается через $y_0(t)$ следующим образом:

$$y(t) = y_0(t) + \frac{b_1}{b_0} \frac{dy_0}{dt} + \frac{b_2}{b_0} \frac{d^2 y_0}{dt^2} + \dots + \frac{b_m}{b_0} \frac{d^m y_0}{dt^m}.$$

В качестве компонент вектора состояния $\|s(t)\|$ выбраны нормированные множителем a_n/b_0 сигнал $y_0(t)$ и $n-1$ его производных:

$$s_k(t) = \frac{a_n}{b_0} \frac{d^{n-k} y_0}{dt^{n-k}}, \quad k = 1, 2, \dots, n.$$

В этом случае при $m < n$ выражения для параметров системы в переменных состояния имеют следующий вид [4]:

$$\|A\| = \begin{bmatrix} -\frac{a_{n-1}}{a_n} & -\frac{a_{n-2}}{a_n} & \dots & -\frac{a_2}{a_n} & -\frac{a_1}{a_n} & -\frac{a_0}{a_n} \\ 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}, \quad \|B\| = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad (2.15)$$

$$\|C\| = \begin{bmatrix} \frac{b_{n-1}}{a_n} & \frac{b_{n-2}}{a_n} & \dots & \frac{b_1}{a_n} & \frac{b_0}{a_n} \end{bmatrix}, \quad D = 0.$$

В векторе $\|C\|$ компоненты с индексами, превышающими m равны нулю.

Каждое из рассмотренных представлений линейной динамической системы можно использовать для построения разных математических моделей этой системы. Эти модели по-разному реализуются в Simulink (см. приведенные ниже примеры). Выбор конкретной модели определяется, прежде всего, теми задачами, которые ставятся при исследовании, и во-вторых зависит от квалификации и предпочтений исследователя. Чтобы облегчить переход от одного представления линейной системы к другому, в MatLab включен ряд функций преобразования линейных систем:

$[z, p, k] = tf2zp(b, a)$; - преобразует векторы коэффициентов числителя и знаменателя (b, a) в векторы нулей и полюсов (z, p) и коэффициент усиления k ; для обратного преобразования используется функция $[b, a] = zp2tf(z, p, k)$;

$[A, B, C, D] = tf2ss(b, a)$; - преобразует векторы коэффициентов числителя и знаменателя (b, a) в параметры пространства состояний (A, B, C, D) ; для обратного преобразования используется функция $[b, a] = ss2tf(A, B, C, D)$;

$[A, B, C, D] = zp2ss(z, p, k)$; - преобразует векторы нулей и полюсов (z, p) и коэффициент усиления k в параметры пространства состояний (A, B, C, D) ; для обратного преобразования используется функция $[z, p, k] = ss2zp(A, B, C, D)$;

$[r, p, C] = residue(b, a)$; - преобразование передаточной функции из формы (2.5) в форму (2.7); (r, p) – векторы вычетов и полюсов, C – константа не равная нулю, если степени полиномов $B(s)$ и $A(s)$ одинаковы. Эта функция может использоваться и для обратного преобразования $[b, a] = residue(r, p, C)$.

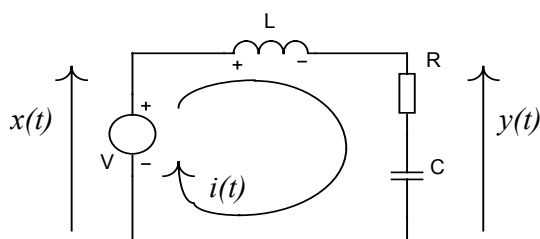


Рис. 2.1

Пример 2.1. Проиллюстрируем рассмотренные выше способы построения математических моделей линейных динамических систем на примере электрической цепи, показанной на рис.2.1. На вход

цепи подается напряжение $x(t)$. Предположим, что ток через индуктивность при включении источника напряжения $i(0)=i_L(0)=0$, а напряжение на конденсаторе $y(0)=V_C(0)=0$ В. В качестве реакции системы будем рассматривать напряжение на конденсаторе в произвольные моменты времени. Для верификации моделей будем использовать переходную характеристику $g(t)$, т.е. ее реакцию на единичный скачок напряжения $x(t)=u_0(t)$. Построим математические модели рассматриваемой линейной системы рассмотренными выше способами.

Построим математическую модель в форме операторного коэффициента передачи (2.5). Для этого воспользуемся операторным методом анализа, заменим в цепи (рис. 2.1) реактивные компоненты их операторными импедансами: $X_L = sL$, $X_C = 1/sC$ и представим цепь в форме комплексных сопротивлений (рис. 2.2).

Операторный коэффициент передачи этой цепи

$$\begin{aligned} K(s) &= \frac{y(s)}{x(s)} = \frac{i(s) \cdot (R + X_C)}{i(s) \cdot (R + X_L + X_C)} = \frac{(R + X_C)}{R + X_L + X_C} = \\ &= \frac{R + 1/sC}{R + sL + 1/sC} = \frac{sRC + 1}{s^2 LC + sRC + 1}. \end{aligned} \quad (2.16)$$

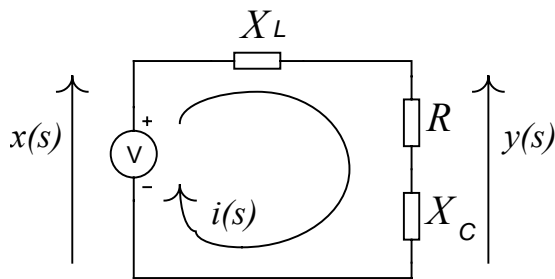


Рис. 2.2

Из описания системы в форме (2.16) нетрудно дифференциальное уравнение исследуемой системы. Это удобно сделать, воспользовавшись правилами операционного исчисления. Их выражения (2.16) имеют

$$y(s) = K(s) \cdot x(s) = \frac{s \cdot RC + 1}{s^2 \cdot LC + s \cdot RC + 1},$$

откуда следует что

$$LC \cdot s^2 \cdot y(s) + RC \cdot s \cdot y(s) + y(s) = RC \cdot s \cdot x(s) + x(s). \quad (2.17)$$

Поскольку по правилам операционного исчисления умножение изображения на s^k соответствует k – кратному дифференцированию оригинала, то из (2.17) получаем следующее дифференциальное уравнение:

$$LC \cdot \frac{d^2 y}{dt^2} + RC \cdot \frac{dy}{dt} + y(t) = RC \cdot \frac{dx}{dt} + x(t). \quad (2.18)$$

Из выражений (2.16), (2.18) найдем векторы $\mathbf{b} = [RC, 1]$ и $\mathbf{a} = [LC, RC, 1]$.

Для построения моделей системы в формах (2.6, 1.7, 1.11) воспользуемся соответствующими функциями MatLab (см. стр. 40). Ниже в качестве примеров приведены листинги программ и результаты расчетов в MatLab нулей и полюсов передаточной функции рассмотренной в примере 2.1 системы и ее переменных состояния

Листинг 1. Расчет нулей и полюсов передаточной функции

```
>>% R=0,75 Ом, L=0,25 Гн, C=2 Ф.      z = -0.6667
>> a=[0.5, 1.5, 1];                  p = -2
>> b=[1.5, 1];                        -1
>> [z, p, k] = tf2zp (b, a)          k = 3
```

Листинг 2 Расчет переменных состояния

```
>>% R=0,75 Ом, L=0,25 Гн, C=2 Ф.      A = -3 -2
>> a=[0.5, 1.5, 1];                  1 0
>> b=[1.5, 1];                      B = 1
>> [A, B, C, D] = tf2ss (b, a)        0
                                         C = 3 2
                                         D = 0
```

Для нахождения импульсной характеристики системы представим ее передаточную функцию в форме (2.7). Применив функцию $[r, p, C] = \text{residue}(b, a)$ для векторов $a = [0.5, 1.5, 1]$ и $b = [1.5, 1]$ найдем полюса и вычеты системы: $r_1 = 4$; $r_2 = -1$; $p_1 = -2$; $p_2 = -1$; $C = 0$. Таким образом,

$$K(s) = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} = \frac{4}{s + 2} - \frac{1}{s + 1}. \quad (2.19)$$

Сложив между собой две последние дроби, получим операторный коэффициент передачи в форме (2.5):

$$K(s) = \frac{1,5s+1}{0,5s^2+1,5s+1}.$$

Следовательно, $b=[1.5, 1]$, $a=[0.5, 1.5, 1]$, что подтверждает правильность выполненного преобразования.

Поскольку среди полюсов нет кратных, то, воспользовавшись формулой (2.8), найдем импульсную характеристику системы:

$$h(t) = r_1 \exp(p_1 t) + r_2 \exp(p_2 t) = 4 \exp(-2t) - \exp(-t).$$

На рис 2.3 показана программа для расчета импульсной характеристики $h(t)$ MatLab и ее график, рассчитанный по этой программе. На рис. 4-6 приведены модели рассматриваемой системы, представленной в формах (2.5), (2.6) и (2.11), и результаты моделирования, когда на ее вход подается короткий прямоугольный импульс, длительностью 0,01с и амплитудой 100В, начинающийся в момент времени $t=1$ с.

```
>> % Расчет импульсной
>> % характеристики
>> t=0:0.01:10;
>> y=4*exp(-2*t)-exp(-t);
>> plot(t,y), grid
```

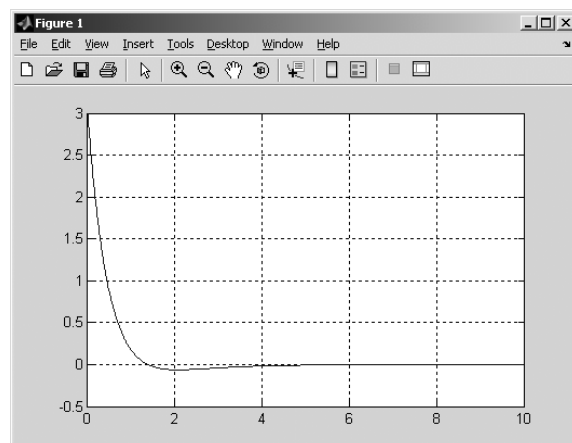


Рис. 2.3

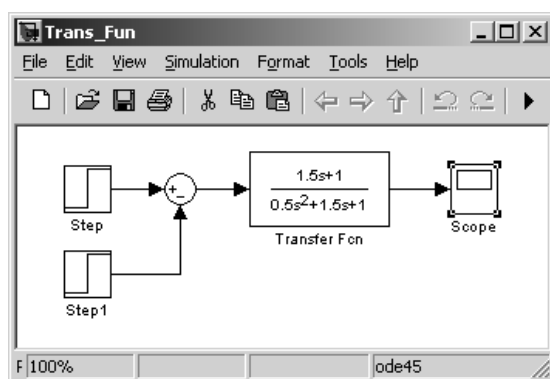


Рис. 2.4

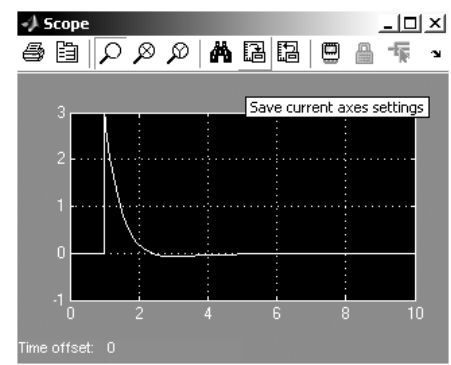
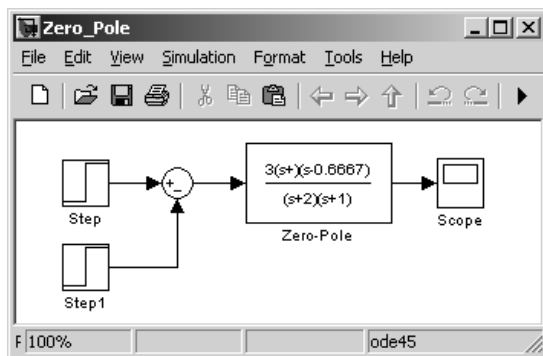


Рис. 2.5

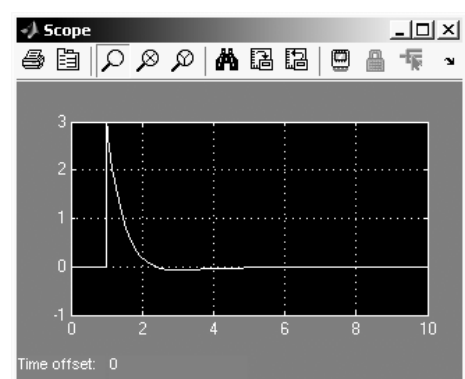
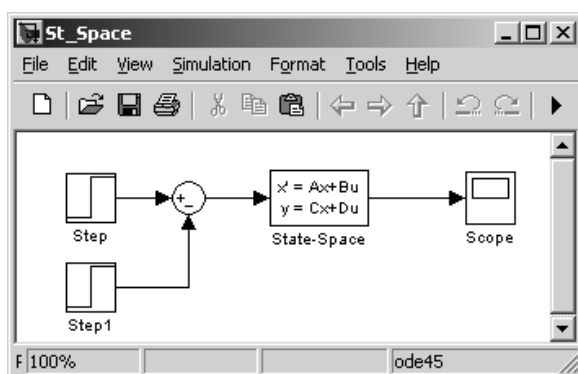


Рис. 2.6

При построении математических моделей использовались параметры системы, приведенные и рассчитанные в примере 2.1. Прямоугольный импульс моделировался вычитанием друг из друга двух скачков напряжения величиной 100В, сдвинутых во времени друг относительно друга на 0,01с. Из анализа результатов моделирования можно сделать вывод о том, что они не зависят от вида математической модели системы.

Пример 2.2. Пусть исследуемая система описывается дифференциальным уравнением четвертого порядка

$$\frac{d^4 y}{dt^4} + a_3 \frac{d^3 y}{dt^3} + a_2 \frac{d^2 y}{dt^2} + a_1 \frac{dy}{dt} + a_0 y(t) = x(t), \quad (2.20)$$

где $y(t)$ – реакция системы (например, выходное напряжение или ток) на входное воздействие $x(t)$. Начальные условия – положим нулевыми: $y(0) = y'(0) = y''(0) = y'''(0) = 0$.

Построим математическую модель системы в форме переменных состояния. При верификации модели используем какой-либо из известных результатов, например: решение дифференциального уравнения

$$\frac{d^4 y}{dt^4} + 2 \frac{d^2 y}{dt^2} + y(t) = \sin t,$$

с начальными условиями $y(0) = y'(0) = y''(0) = y'''(0) = 0$, имеет вид

$$y(t) = \frac{(3 - 3t - t^2) \cos t}{8}.$$

Для верификации модели мы рассчитаем в MatLab по этой формуле $y(t)$, а в Simulink выберем коэффициенты $a_4=1$, $a_3=0$, $a_2=2$, $a_1=0$ и $a_0=1$, в качестве входного сигнала используем гармонический сигнал с частотой $1/2\pi$ и сравним отклики: рассчитанный в MatLab и полученный в Simulink.

Дифференциальное уравнение (2.20) имеет четвертый порядок, следовательно, необходимо определить четыре переменных состояния, которые будут использованы четырьмя уравнениями состояния первого порядка.

Выберем в качестве переменных $s_1(t)$, $s_2(t)$, $s_3(t)$ и $s_4(t)$ реакцию системы $y(t)$ и ее производные:

$$s_1(t)=y(t), \quad s_2(t)=\frac{dy}{dt}, \quad s_3(t)=\frac{d^2 y}{dt^2}, \quad s_4(t)=\frac{d^3 y}{dt^3}$$

и выразим через них дифференциальное уравнение (2.20) в виде системы уравнений 4-го порядка:

$$\frac{ds_1}{dt} = s_2(t);$$

$$\frac{ds_2}{dt} = s_3(t);$$

$$\frac{ds_3}{dt} = s_4(t);$$

$$\frac{ds_4}{dt} = -a_0 s_1(t) - a_1 s_2(t) - a_2 s_3(t) - a_3 s_4(t) + x(t);$$

Запишем эту систему в матричной форме

$$\begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \dot{s}_3 \\ \dot{s}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} x(t)$$

или более компактно

$$\|\dot{s}\| = \|A\| \|s\| + \|B\| \|x\|$$

Выходной сигнал в соответствии с формулой (2.15) определяется выражением

$$y(t) = \|C\| \|s\| + D \|x\| = [1 \ 0 \ 0 \ 0] \cdot \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} + 0 \cdot [0 \ 0 \ 0 \ x] = s_1(t).$$

В соответствии с изложенным выше системы положим $a_3 = 0$, $a_2 = 2$, $a_1 = 0$, $a_0 = 1$ и $x(t) = \sin(t)$, следовательно, параметры системы в рассматриваемом пространстве состояний будут следующими:

$$\|A\| = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -2 & 0 \end{bmatrix}, \quad \|B\| = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\|C\| = [1 \ 0 \ 0 \ 0], \quad D = 0; \quad x(t) = \sin(t);$$

На рис. 2.7 показаны команды MatLab и результаты расчета решения дифференциального уравнения (2.20), которое представляет отклик исследуемой системы на внешнее воздействие вида $x(t) = \sin(t)$

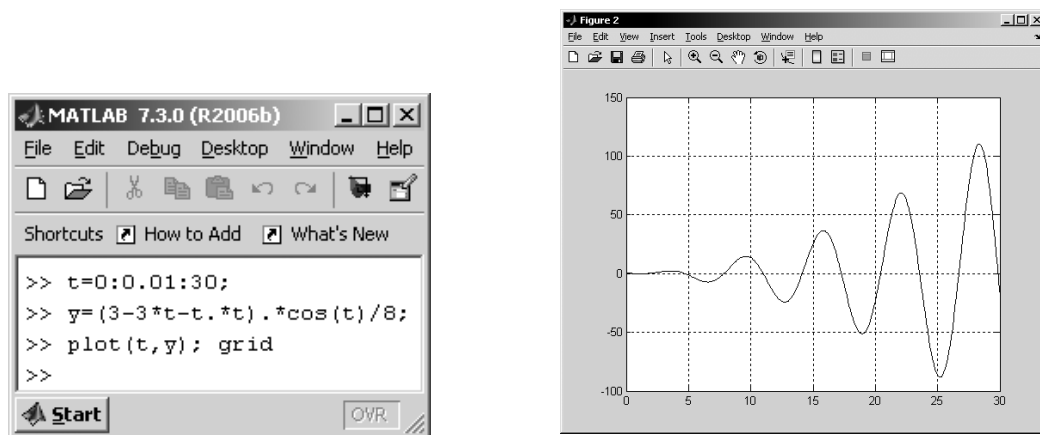


Рис. 2.7

На рис. 2.8 и 2.9 приведены результаты моделирования исследуемой системы в Simulink, когда модель системы построена непосредственно по дифференциальному уравнению (рис. 2.8) и в форме переменных состояния (рис. 2.9).

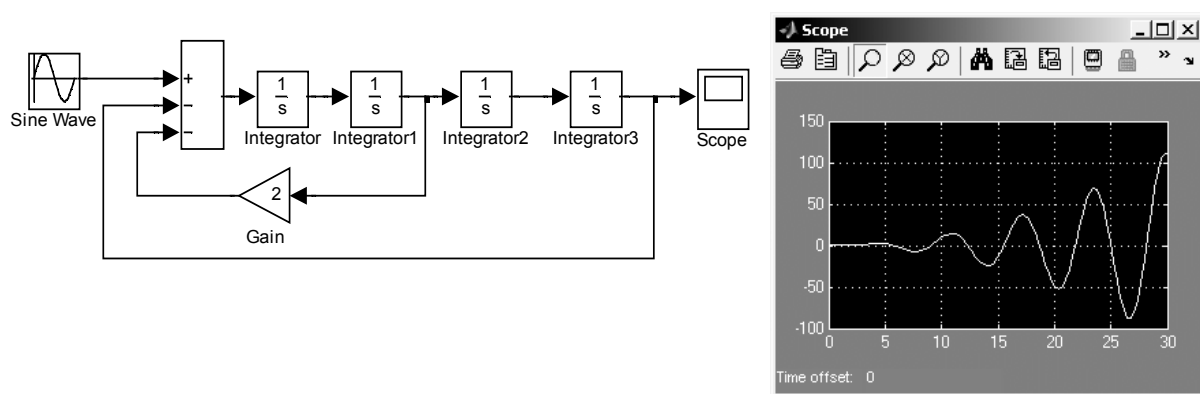


Рис. 2.8

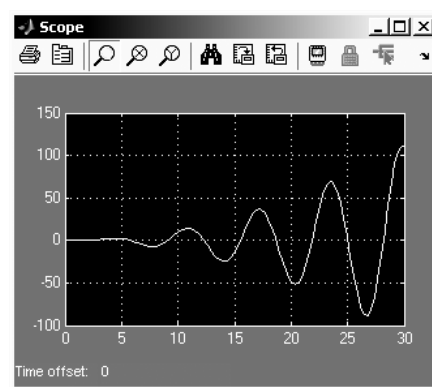
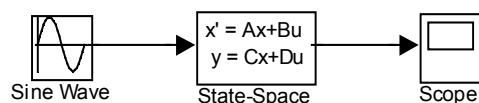
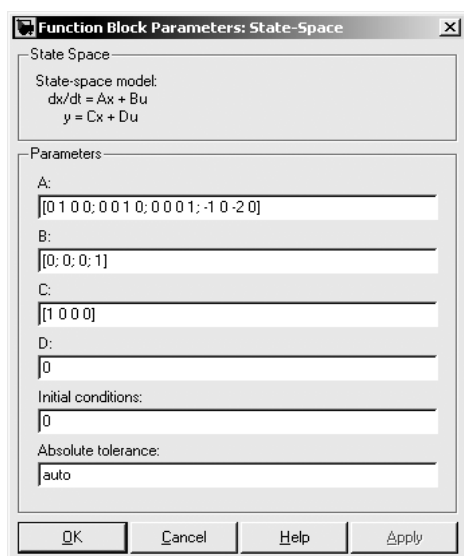


Рис. 2.9

Сопоставляя результаты моделирования и результаты расчетов можно сделать следующие выводы:

- 1) результаты моделирования системы в форме дифференциального уравнения и в форме переменных состояния совпали;
- 2) результаты моделирования системы несколько отличаются от теоретического решения, в частности временным масштабом.

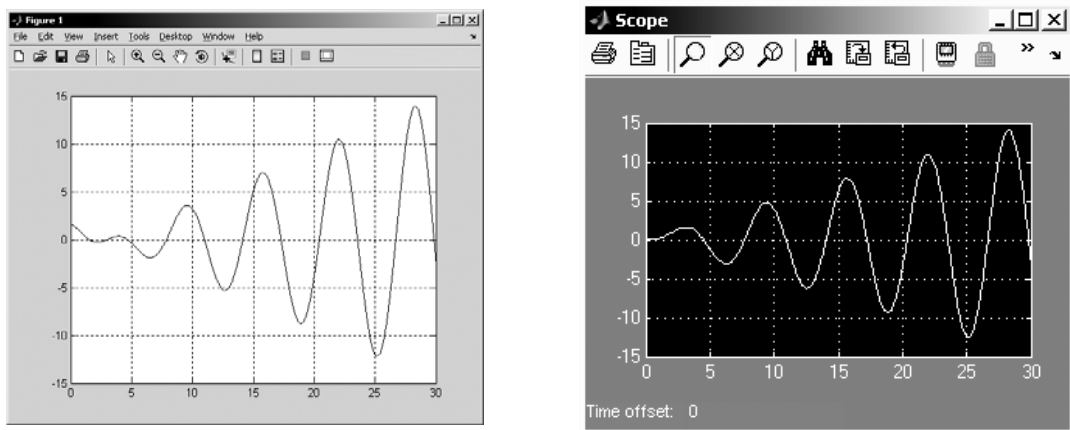


Рис. 2.10

Вместе с тем импульсная характеристика исследуемой системы, полученная на рассмотренных моделях практически не отличается от теоретической (рис. 2.10). Это обстоятельство, по-видимому, можно объяснить достаточно большими ошибками вычислительных схем MatLab при их старте в случае малых возмущающих воздействий.

2.3. Методы описания дискретных линейных систем

Дискретную динамическую систему подобно непрерывной динамической системе называют линейной и стационарной, если она обладает свойствами линейности и стационарности:

- а) реакция системы на сумму нескольких внешних воздействий равна сумме реакций на каждое из этих воздействий, взятых по отдельности;

б) задержка входного воздействия приводит лишь к такой же задержке реакции системы без изменения ее формы, т. е. Параметры стационарной системы во времени не изменяются.

Все операции, выполняемые линейными стационарными системами, ограничиваются сложением и умножением на константы. Сущность линейной дискретной обработки можно пояснить следующими примерами [6]. Рассмотрим дискретные системы, реакция которых $y(k)$ на воздействия $x(k)$ описывается следующими выражениями:

$$\begin{aligned}y_a(k) &= x(k) + x(k-1); \\y_b(k) &= x(k) + y_b(k-1); \\y_c(k) &= x(k) + 0,5y_c(k-1).\end{aligned}$$

В общем случае дискретная линейная стационарная система суммирует с некоторыми весами несколько отсчетов входного сигнала $x(k)$ и некоторое количество предыдущих отсчетов выходного сигнала $y(k-1-i)$:

$$\begin{aligned}y(k) &= b_0 x(k) + b_1 x(k-1) + \dots + b_m x(k-m) - \\&\quad a_1 y_c(k-1) - a_2 y_c(k-2) - a_n y_c(k-n).\end{aligned}\quad (2.21)$$

Эта формула определяет алгоритм формирования отклика (реакции) $y(k)$ рассматриваемой системы на входное воздействие $x(k)$.

Если отсчеты выходного и входного сигнала сосредоточить соответственно в левой и правой частях уравнения, то последнюю формулу можно переписать в следующем виде:

$$\begin{aligned}y(k) + a_1 y(k-1) + a_2 y(k-2) + a_n y(k-n) &= \\b_0 x(k) + b_1 x(k-1) + \dots + b_m x(k-m).\end{aligned}\quad (2.21)$$

Эту форму записи называют разностным уравнением. Структура разностного уравнения для дискретной линейной стационарной системы сходна со структурой дифференциального уравнения для непрерывной линейной стационарной системой (см. выражение (2.10)). Поэтому и формы описания таких систем имеют много общего.

Математическую модель дискретной линейной стационарной системы можно представить в одной из следующих форм:

- 1) с помощью импульсной характеристики;

- 2) с помощью переходной характеристики;
- 3) с помощью функции передачи (в полиномиальной или дробно-рациональной форме, набором нулей и полюсов или набором полюсов и вычетов);
- 4) с помощью комплексного коэффициента передачи (частотной характеристики);
- 5) с помощью разностного уравнения;
- 6) с помощью вектора состояния системы в некотором пространстве состояний (state space).

Импульсной характеристикой $h(k)$ дискретной линейной системы называют ее реакцию на единичную импульсную функцию

$$x_0(k) = \begin{cases} 1 & \text{при } k = 0, \\ 0 & \text{при } k \neq 0. \end{cases}$$

$x_0(k)$ для дискретной системы является аналогом δ – функции для непрерывной системы.

Переходной характеристикой $g(k)$ дискретной линейной системы называют ее реакцию на бесконечную последовательность единичных отсчетов

$$u_0(k) = \begin{cases} 1 & \text{при } k \geq 0, \\ 0 & \text{при } k < 0. \end{cases}$$

Реакция дискретной линейной системы на произвольное входное воздействие определяется формулой дискретной свертки (2.22).

$$y(k) = \sum_{m=-\infty}^k x(m) h(k-m). \quad (2.22)$$

Функция передачи (системная функция) $H(z)$ дискретной линейной системы является аналогом операторного коэффициента передачи $K(s)$ непрерывной линейной системы и подобно ему может быть выражена через импульсную характеристику системы:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{\infty} h(k) z^{-k}, \quad (2.23)$$

где $X(z)$ и $Y(z)$ – z -образы входного сигнала и реакции системы;

$z = \exp(sT)$, $s = \sigma + j\omega$ – лапласовская переменная;

T – интервал дискретизации сигнала.

Функция передачи дискретной линейной системы, описываемой разностным уравнением (2.21), имеет следующий вид:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_n z^{-n}}. \quad (2.24)$$

Подобно операторному коэффициенту передачи непрерывной линейной системы (2.5) функция передачи дискретной системы (2.24) может быть представлена набором нулей и полюсов

$$H(z) = \frac{(1 - z_1 z^{-1})(1 - z_2 z^{-1}) \dots (1 - z_m z^{-1})}{(1 - p_1 z^{-1})(1 - p_2 z^{-1}) \dots (1 - p_n z^{-1})}, \quad (2.25)$$

а также набором полюсов и вычетов

$$H(z) = \frac{r_1}{(1 - p_1 z^{-1})} + \frac{r_2}{(1 - p_2 z^{-1})} + \dots + \frac{r_n}{(1 - p_n z^{-1})} + \\ + C_0 + C_1 z^{-1} + \dots + C_{m-n} z^{-(m-n)}. \quad (2.26)$$

В формулах (2.25, 2.26)

$z_1, z_2 \dots z_m$ – нули функции передачи $H(z)$ (корни полинома в числителе выражения (2.24);

$p_1, p_2 \dots p_n$ – полюсы функции передачи $H(z)$ (корни полинома в знаменателе выражения (2.24);

$r_1, r_2 \dots r_m$ – вычеты функции передачи $H(z)$ относительно соответствующих полюсов;

C_0, C_1, C_{m-n} – константы, отличные от нуля при $m \geq n$.

Полюсы и вычеты функции передачи могут быть действительными или комплексно-сопряженными числами. Паре комплексно-сопряженных полюсов соответствует пара комплексно-сопряженных вычетов. Если в формулах (2.25) и (2.26) объединить друг с другом сомножители и слагаемые с комплексно сопряженными полюсами, то нетрудно заметить, что выражение (2.25) можно трактовать как произведение, а выражение (2.26) как сумму передаточных функций первого второго порядков с действительными коэффициентами.

Из формул (2.24) - (2.26) следует, что дискретную линейную систему можно реализовать различными способами [6]:

- 1) в прямой форме, реализующей в различных вариантах (каноническом, транспонированном и т.д.) алгоритм (2.21);
- 2) в виде последовательной цепочки звеньев первого и второго порядка;
- 3) в виде параллельного соединения звеньев первого и второго порядков.

При отсутствии среди p_i кратных полюсов импульсная характеристика дискретной системы представляет собой сумму откликов $h_i(k)$ дискретных звеньев первого порядка с функциями передачи

$$H_i(z) = \frac{r_i}{(1 - p_i z^{-1})}, \quad (i = 1, \dots, n).$$

Отклик каждого из таких звеньев имеет следующий вид:

$$h_i(k) = \sum_{k=0}^{\infty} r_i p_i^k. \quad (2.27)$$

Если полюсы и вычеты суть числа комплексно-сопряженные, то каждой такой паре соответствует отклик

$$h_i(k) = \sum_{k=0}^{\infty} r_i p_i^k + r_i^* p_i^{*k} = 2|r_i||p_i|^k \cos(k\varphi_{p_i} + \varphi_{r_i}), \quad (2.27)$$

где φ_{p_i} , φ_{r_i} — фазы комплексных чисел p_i и r_i .

Такой паре полюсов и вычетов соответствует звено второго порядка с функцией передачи

$$H_i(z) = \frac{r_i^* r_i}{(1 - p_i z^{-1})(1 - p_i^* z^{-1})} = \frac{|r_i|^2}{(1 - 2\operatorname{Re}(p_i)z^{-1} + 2|p_i|^2 z^{-2})}.$$

Из формул (2.26, 2.27) следует, что условием устойчивости дискретной линейной системы является выполнение системы неравенств

$$|p_i| < 1, \quad (i = 1, \dots, n),$$

т.е. все полюса функции передачи должны лежать внутри круга единичного радиуса - в этом случае все слагаемые импульсной характеристики с ростом k будут убывать.

Комплексный коэффициент передачи (частотную характеристику дискретной линейной системы) можно выразить через функцию передачи, заменив в ней z^{-1} на $\exp(j\omega T)$.

$$\dot{K}(\omega) = H(\exp(j\omega T)) = \sum_{k=0}^{\infty} h(k) \exp(-j\omega kT). \quad (2.28)$$

Подобно непрерывной линейной системе дискретная линейная система может быть описана некоторым вектором $\|s(k)\|$ в пространстве состояний.

$$\begin{aligned} \|s(k+1)\| &= \|A\| \|s(k)\| + \|B\| x(k) \\ y(k) &= \|C\| \|s(k)\| + D x(k). \end{aligned} \quad (2.29)$$

Здесь $\|A\|$, $\|B\|$, $\|C\|$, D - параметры, описывающие систему. Для скалярных сигналов $x(k)$ и $y(k)$ и размерности вектора состояний $\|s(k)\|$, равной N , $\|A\|$ - матрица $N \times N$, $\|B\|$ - столбец $N \times 1$, $\|C\|$ - строка $1 \times N$, D - скаляр. Как и в случае непрерывных систем, преобразование коэффициентов функции передачи в параметры пространства состояний не является однозначным. Оно определяется выбором вектора $\|s(k)\|$.

Для преобразования способов описаний дискретных систем в MatLab используются описанные выше (стр. 40) функции.

$[z, p, k] = tf2zp(b, a)$; - преобразует векторы коэффициентов числителя и знаменателя (b, a) в векторы нулей и полюсов (z, p) и коэффициент усиления k ; для обратного преобразования используется функция $[b, a] = zp2tf(z, p, k)$;

$[A, B, C, D] = tf2ss(b, a)$; - преобразует векторы коэффициентов числителя и знаменателя (b, a) в параметры пространства состояний (A, B, C, D) ; для обратного преобразования используется функция $[b, a] = ss2tf(A, B, C, D)$;

$[A, B, C, D] = zp2ss(z, p, k)$; - преобразует векторы нулей и полюсов (z, p) и коэффициент усиления k в параметры пространства состояний (A, B, C, D) ; для обратного преобразования используется функция $[z, p, k] = ss2zp(A, B, C, D)$.

Особенностью применения этих функций к дискретным системам является то, что множества (векторы) коэффициентов полиномов числителя и знаменателя \mathbf{b} и \mathbf{a} функции передачи $H(z)$ в MatLab должны иметь одинаковую длину. Поэтому применение любого из указанных преобразований необходимо предварять вызовом функции $[b1, a1] = \text{eqtflength}(b, a)$, которая дополнит более короткий из векторов \mathbf{b} или \mathbf{a} нулями и преобразует их в вектора $\mathbf{b1}$ и $\mathbf{a1}$.

Для преобразования функции передачи дискретной системы из формы (2.24) в форму (2.26) используется функция $[r, p, C] = \text{residuez}(b, a)$. Эта функция отличается от соответствующей функции $\text{residue}(b, a)$ для преобразования непрерывных систем как формой представления результатов (сравните формулы (2.7) и (2.26)). Функция residuez подобно функции residue позволяет выполнять и обратное преобразование: $[b, a] = \text{residuez}(r, p, C)$.

Пример 2.3. Проиллюстрируем некоторые методы построения и преобразования моделей дискретных линейных систем на примере моделирования линейной системы третьего порядка с функцией передачи (2.24), заданной векторами \mathbf{b} и \mathbf{a} , компоненты которых суть коэффициенты полиномов числителя знаменателя функции передачи (2.24): $\mathbf{b} = \{b_0, b_1, b_2, b_3\}$, $\mathbf{a} = \{a_0, a_1, a_2, a_3\}$. В качестве такой линейной системы выберем баттервортовский фильтр нижних частот 3-го порядка. Для расчета коэффициентов его функции передачи воспользуемся функцией MatLab

$$[b, a] = \text{butter}(n, v_c),$$

где n – порядок фильтра;

$v_c = 2 * f_c * T$ – частота среза фильтра в Гц, нормированная по частоте дискретизации входного сигнала.

Результаты расчета векторов \mathbf{b} и \mathbf{a} будут выведены в командное окно MatLab и одновременно размещены в области памяти, называемой рабочей областью (workspace) под именами \mathbf{b} и \mathbf{a} . Значения переменных \mathbf{b} и \mathbf{a} доступны для дальнейших расчетов и будут храниться в рабочей области, пока пользователь не удалит их оттуда, либо не заменит другими значениями с теми же именами.

На рис. 2.11 показаны результаты расчета в MatLab векторов \mathbf{b} и \mathbf{a} . Функция $\text{impz}(b, a, 30)$ рассчитывает и выводит график импульсной характеристики дискретной системы, задаваемой векторами \mathbf{b} и \mathbf{a} в 30

точках. Если результаты расчета необходимо сохранить для дальнейшего использования функция используется в формате $[h,t]=\text{impz}(b,a,30)$. В этом случае в рабочую область будут выведены массивы отсчетов импульсной характеристики h и нормированных по частоте дискретизации моментов времени t , соответствующих отсчетам импульсной характеристики.

```
>> [b,a]=butter(3,0.2)
b =
    0.0181    0.0543    0.0543    0.0181
a =
    1.0000   -1.7600    1.1829   -0.2781
>> impz(b,a,30)
```

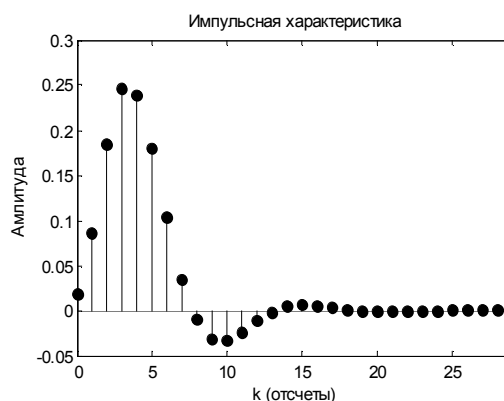


Рис. 2.11

На рис. 2.12 показана математическая модель рассматриваемой дискретной системы, заданной функцией передачи в форме (2.24) и окно установки параметров моделирования.

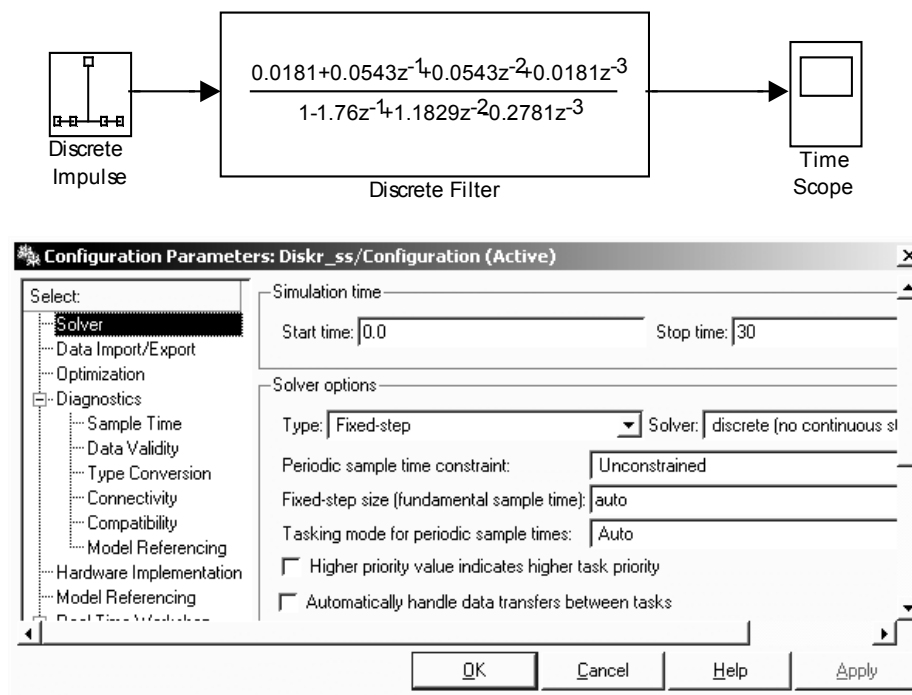


Рис. 2.12

Компонент Discrete Impulse (Simulink|Signal Processing Blockset|Signal Processing Sources) моделирует единичную импульсную функцию $x_0(k)$, Discrete Filter (Simulink|Discrete) – модель рассматриваемой дискретной системы, Time Scope (Simulink|Signal Processing Blockset|Signal Processing Sinks) – виртуальный осциллограф для наблюдения за результатами моделирования.

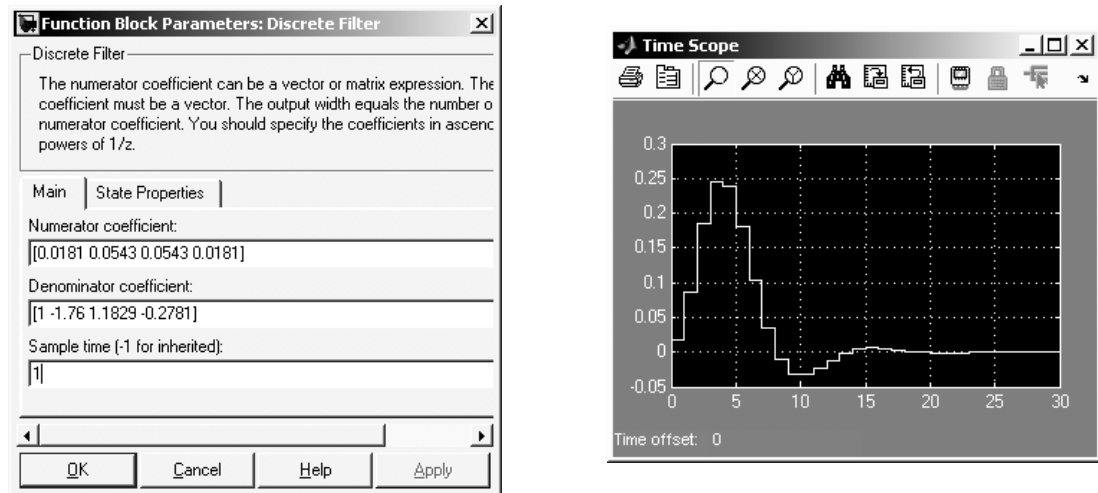


Рис. 2.13

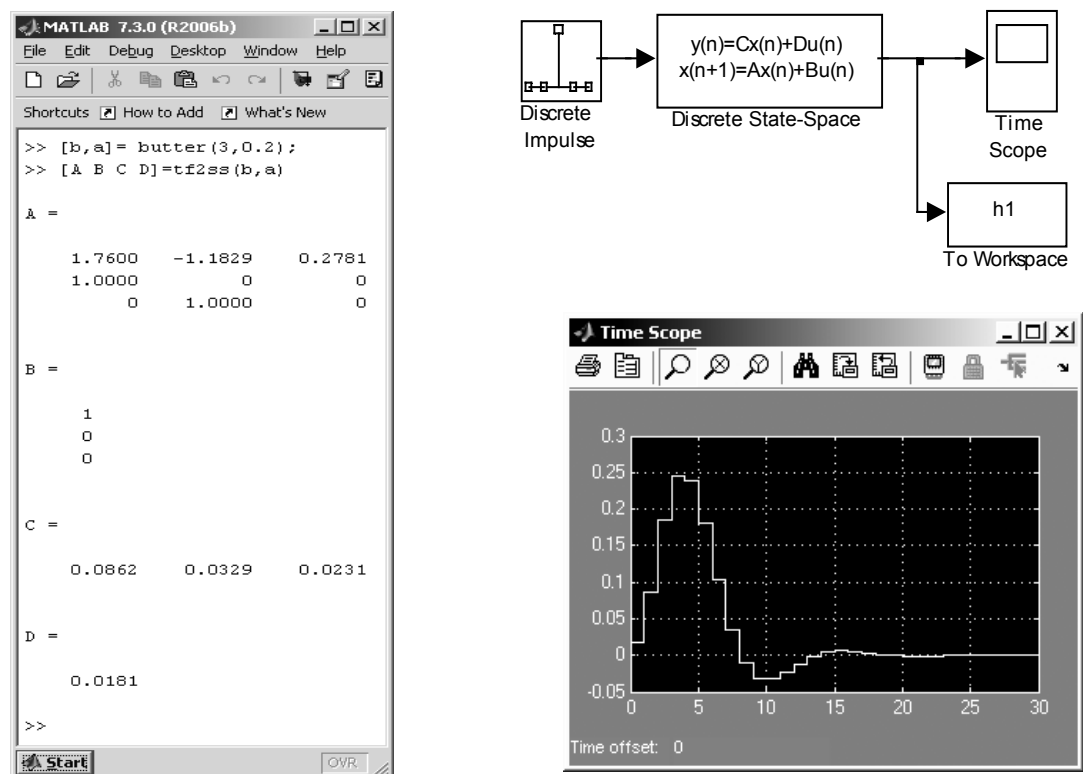


Рис. 2.14

При моделировании дискретных систем необходимо соответствующим образом изменить установки подсистемы Solver Simulink в окне Configuration Parameters. Это окно активизируется командой Simulation| Configuration Parameters...(Ctrl+E).

На рис. 2.13 приведены окно установки параметров блока Discrete Filter и результаты его моделирования – отклик на воздействие единичной импульсной функции $x_0(k)$, т.е. импульсная характеристика исследуемой дискретной системы. Параметры системы – вектора \mathbf{b} и \mathbf{a} – вводятся соответственно в поля Numerator Coefficient и Denominator Coefficient окна Function Block Parameters.

На рис. 2.14 показано командное окно MatLab для расчета параметров математической модели рассматриваемой дискретной системы в форме пространства состояний (2.29) и приведены результаты моделирования импульсной характеристики системы.

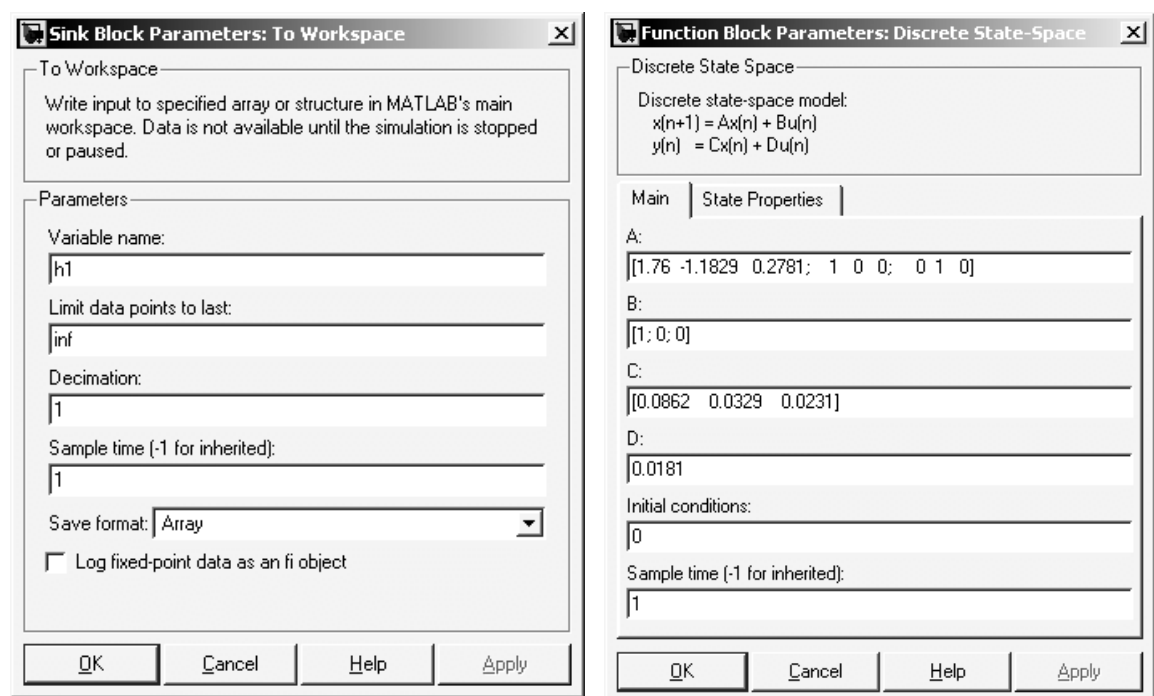


Рис. 2.15

На рис. 2.15 показаны окна установки параметров этих блоков. Результаты моделирования отображаются виртуальным осциллографом Time Scope. Компонент Discrete State-Space (Simulink| Discrete) –

модель рассматриваемой дискретной системы в пространстве состояний, блок to Workspace используется для вывода результатов моделирования с выхода блока Discrete State-Space в рабочую область MatLab (рис. 2.15) в форме массива чисел под именем h1.

Сопоставление результатов моделирования (рис. 2.11, 2.13, 2.14) позволяет сделать вывод о том независимо от выбора вида дискретной математической модели результаты моделирования не изменяются. Следовательно, с точки зрения поведения системы эти модели эквивалентны друг другу.

Дискретные модели непрерывных систем

Рассмотрим алгоритмы преобразования моделей систем, заданных в виде дифференциальных уравнений, к разностным уравнениям. Это преобразование связано с задачей построения *дискретных моделей непрерывных систем*. Такая задача ставится следующим образом. Пусть математическая модель непрерывной системы имеет вид

$$\begin{aligned}\dot{s}(t) &= \|A\|s(t) + \|B\|x(t) \\ y(t) &= \|C\|s(t) + Dx(t).\end{aligned}$$

Требуется получить модель эквивалентной дискретной системы в виде разностных уравнений

$$\begin{aligned}s(k+1) &= \|A_{\Delta}\|s(k) + \|B_{\Delta}\|x(k) \\ y(k) &= \|C_{\Delta}\|s(k) + D_{\Delta}x(k).\end{aligned}$$

Системы будем считать эквивалентными, если при соответствующих начальных условиях их реакции на одно и то же входное воздействие совпадают, т.е. при $x(k) = x(t_k)$ выполняется равенство $y(k) = y(t_k)$. Здесь $x(k)$ и $y(k)$ входное воздействие и отклик в k -м отсчете для дискретной системы, $x(t_k)$ и $y(t_k)$ входное воздействие и отклик в момент времени $t_k = kT$ для непрерывной системы, T – интервал дискретизации.

Необходимость перехода от модели непрерывной системы к модели дискретной системы позволяет существенно упростить решение ряда задач [4]:

- 1) исследования импульсных систем;
- 2) исследование цифровых систем управления
- 3) синтез цифровых систем управления по непрерывным моделям и т. д.

В общем случае поставленная задача точного решения не имеет. Это связано с тем, что при дискретизации входного процесса теряется информация между отсчетными точками $t_k = kT$, $k=0, 1, 2, \dots$.

Следовательно, поведение дискретной модели от изменений входного процесса между отсчетными точками не зависит, тогда как реакция непрерывной системы зависит от всех значений входного процесса. Однако, если последовательность отсчетов $x(k) = x(t_k)$ однозначно определяет входной процесс $x(t)$ на интервале моделирования, возможно точное решение задачи. Эта ситуация характерна для цифровых систем передачи непрерывных сообщений с ограниченным спектром, цифровых систем управления, когда управляющие воздействия дискретны по своей природе и ряда других задач.

В таких задачах применяют ступенчатую экстраполяцию входного процесса, т.е. полагают, что он не изменяется между отсчетными точками: $x(k) = x(t_k) = x(kT)$, $k=0, 1, 2, \dots$.

В классической теории управления задача построения дискретной модели непрерывной системы решается на основе аппарата передаточных функций и z -преобразования. Функция передачи $H(z)$ дискретной модели связана с операторным коэффициентом передачи $K(s)$ следующим образом:

$$H(z) = (1 - z^{-1}) Z\left(\frac{K(s)}{s}\right),$$

где $Z(K(s)/s)$ – z -преобразование функции $K(s)/s$.

От функции $H(z)$ можно перейти к модели дискретной системы в форме разностного уравнения (см. формулу 2.21).

Решение рассматриваемой задачи на основе метода пространства состояний имеет следующий вид:

$$H(z) = \|C_{\Delta}\| \left(z \|I_n\| - \|A_{\Delta}\| \right)^{-1} \|B_{\Delta}\| + D_{\Delta}.$$

Параметры модели пространства состояний эквивалентной дискретной системы $\|A_{\Delta}\|, \|C_{\Delta}\|, \|B_{\Delta}\|, D_{\Delta}$ определяются следующими формулами:

$$\begin{aligned} \|A_{\Delta}\| &= \exp(\|A\|T); \quad \|B_{\Delta}\| = \|A\|^{-1} (\|A_{\Delta}\| - \|I_n\|) \|B\|; \\ \|C_{\Delta}\| &= \|C\|; \quad D_{\Delta} = D. \end{aligned}$$

Приведенными формулами можно пользоваться, когда матрица $\|A\|$ не вырожденная (т.е. ее определитель не равен нулю). Это означает линейную независимость выбранных переменных для представления непрерывной системы в пространстве состояний.

Если это условие не выполняется, для построения модели дискретной системы в пространстве состояний можно пользоваться приближенными формулами либо строить такие модели на основе приводимых ниже подстановочных формул [4].

Для непрерывной системы, заданной в пространстве состояний передаточную функцию эквивалентной дискретной системы можно найти следующим образом:

$$H_{\Pi}(z) = \|C\| \left(\frac{z-1}{T} \|I_n\| - \|A\| \right)^{-1} \|B\| = \|C\| \left(\frac{1-z^{-1}}{z^{-1}T} \|I_n\| - \|A\| \right)^{-1} \|B\|.$$

$$H_{\Pi}(z) = K(s) \Big|_{s = \frac{1-z^{-1}}{z^{-1}T}} = K(s) \Big|_{s = \frac{z-1}{T}}.$$

Эти приближенные формулы получены на основе линейной аппроксимации Паде [4].

Помимо этих формул используют подстановочные формулы, полученные на основе неявного метода Эйлера

$$H_{\Theta}(z) = z^{-1} K(s) \Big|_{s = \frac{1-z^{-1}}{T}} = \frac{1}{z} K(s) \Big|_{s = \frac{z-1}{zT}}$$

и метода Тастина

$$H_T(z) = \frac{2z^{-1}}{1+z^{-1}} K(s) \Big|_s = \frac{2}{T} \frac{1-z^{-1}}{1+z^{-1}} = \frac{2}{z+1} K(s) \Big|_s = \frac{2}{T} \frac{z-1}{z+1}.$$

Точность рассмотренных методов зависит от соотношения между интервалом дискретизации T и наименьшей из постоянных времени динамических звеньев моделируемой системы. При разумном выборе этого соотношения точность оказывается вполне приемлемой. Для цифровых систем управления рекомендуется выбирать интервал дискретизации, не превышающий 5% от времени переходного процесса в системе [4]. Две последние формулы привлекательны еще и тем, что сохраняют устойчивость вычислительной схемы при любых $T > 0$, а не только при его малых значениях.

Для преобразования непрерывной системы в дискретную в MatLab можно применять функцию *c2d*. Ее синтаксис:

$$Hd = c2d(Ks, T, method),$$

где Hd – функция передачи дискретной системы;

Ks – операторный коэффициент передачи непрерывной системы;

T – интервал дискретизации;

method – метод, которым выполняется преобразование:

'zoh' – экстраполяция нулевого порядка;

'foh' – линейная экстраполяция;

'imp' – метод инвариантной импульсной характеристики;

'tustin' – метод Тастина (билинейной аппроксимации).

Функция *c2d* может использоваться с укороченным списком аргументов $Hd = c2d(Ks, T)$ – это эквивалентно записи $Hd = c2d(Ks, T, 'zoh')$. Применению функции *c2d* должно предшествовать применение функции $Ks = tf(b, a)$, которая объединяет векторы \mathbf{b} и \mathbf{a} в передаточную функцию Ks (см. пример 2.3).

Пример 2.4. Построим дискретные модели непрерывной системы, рассмотренной в примере 2.1, операторный коэффициент передачи которой имеет следующий вид:

$$K(s) = \frac{1,5s+1}{0,5s^2 + 1,5s+1}.$$

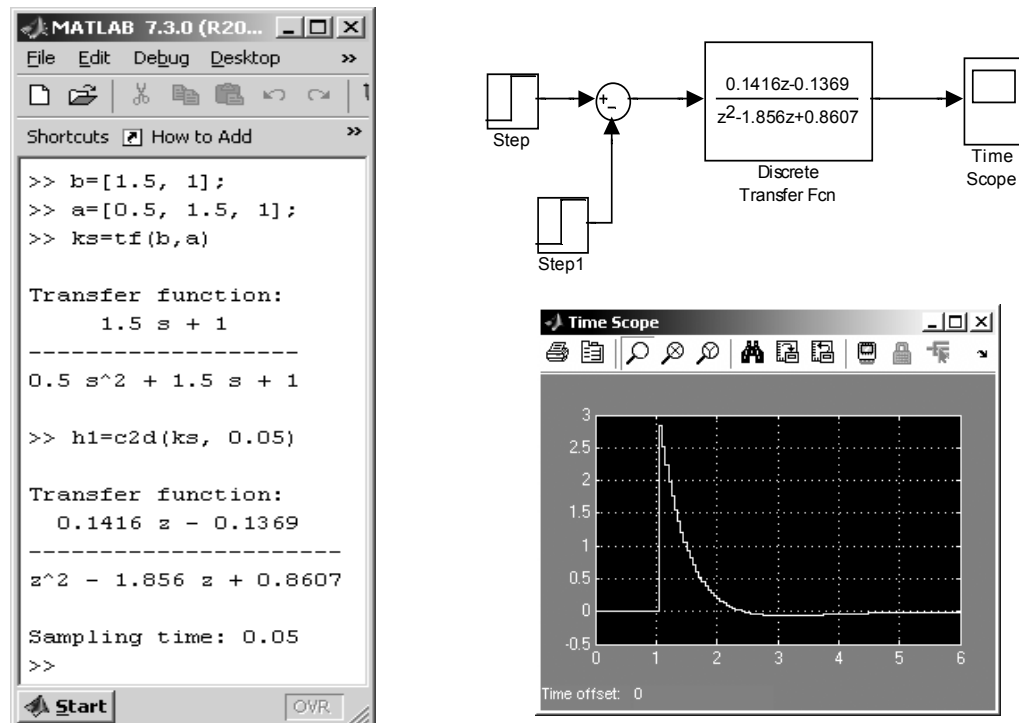


Рис. 2.16

Найдем передаточные функции рассмотренных выше дискретных систем:

$$H_{\Pi}(z) = K(s) \Big|_{s = \frac{z-1}{T}} = \frac{1,5s+1}{0,5s^2 + 1,5s+1} \Big|_{s = \frac{z-1}{T}}$$

$$H_{\Theta}(z) = \frac{1}{z} K(s) \Big|_{s = \frac{z-1}{zT}} = \frac{1}{z} \cdot \frac{1,5s+1}{0,5s^2 + 1,5s+1} \Big|_{s = \frac{z-1}{zT}}$$

$$H_T(z) = \frac{2}{z+1} K(s) \Big|_{s = \frac{2}{T} \frac{z-1}{z+1}} = \frac{2}{z+1} \cdot \frac{1,5s+1}{0,5s^2 + 1,5s+1} \Big|_{s = \frac{2}{T} \frac{z-1}{z+1}}$$

На рис. 2.16 приведены результаты расчета в MatLab передаточной функции дискретной системы при экстраполяции нулевого порядка и интервале дискретизации $T=0.05\text{с}$, математическая модель системы в Simulink и результаты моделирования импульсной реакции

системы. На рис. 2.17 приведены аналогичные результаты для интервалов дискретизации $T=0.25\text{c}$ и $T=0.5\text{c}$.

Сопоставление результатов моделирования с приведенными выше результатами моделирования непрерывной системы (рис. 2.3 – 2.6) позволяет сделать вывод о том, что в рассматриваемом случае даже при интервале дискретизации $T=0,25\text{c}$ (что составляет приблизительно 10% от длительности переходных процессов в системе) дискретную и непрерывную системы вполне можно считать эквивалентными.

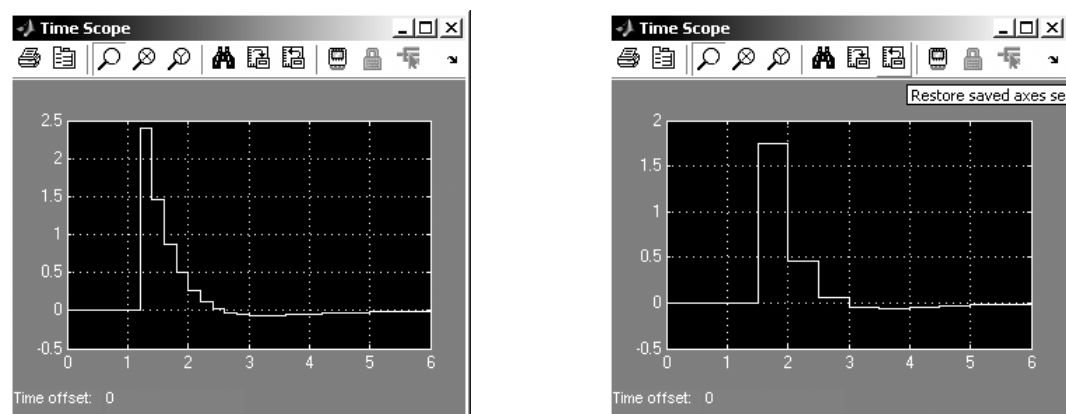


Рис. 2.17

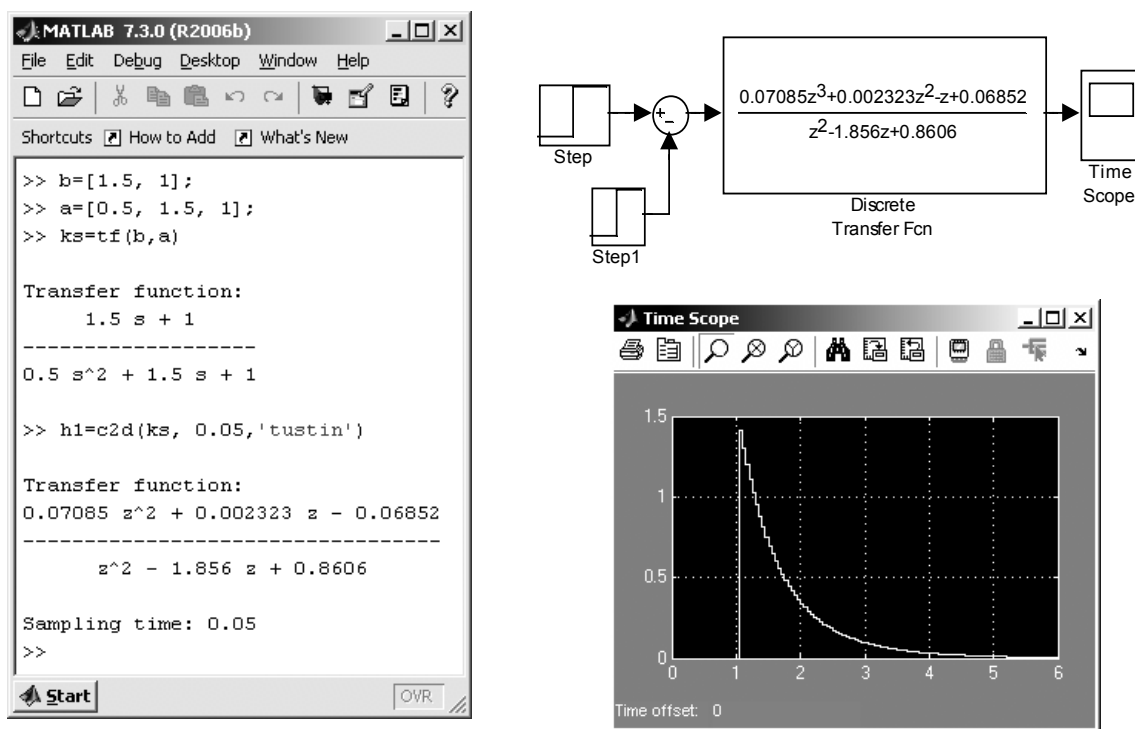


Рис. 2.18

На рис.2.18 приведена аналогичная дискретная модель этой же непрерывной системы построенная методом Тастина (билинейной аппроксимации), расчет параметров этой модели в MatLab и результаты моделирования. Сопоставление импульсных характеристик непрерывной (рис. 2.3 – 2.6) и дискретной (рис. 2.18) систем говорит о том, что во временной области рассматриваемые системы эквивалентными считать нельзя: их импульсные характеристики существенно отличаются даже при интервале дискретизации, равном 2% от длительности переходного процесса в системе. Это подтверждает известный из теории дискретных систем вывод о том, что нельзя построить непрерывную и дискретную системы, которые были бы эквивалентны друг другу и во временной и в частотной областях.

3. БИБЛИОТЕКИ SIMULINK

Характерной особенностью Simulink является наличие обширной библиотеки компонентов. Помимо основной библиотеки, также названной Simulink, и ее дополнения Simulink Extras создано еще около 30 библиотек для приложений Simulink в разных предметных областях и выполнения сервисных функций. При этом с каждой новой версией новой версией Simulink количество компонентов в библиотеках непрерывно увеличивается, а сами библиотеки перекомпоновываются, что, безусловно, затрудняет пользование ими. Вместе с тем в повседневной работе пользователю, обычно необходим достаточно ограниченный набор компонентов. Поэтому представляется целесообразным рекомендовать пользователю организовывать собственные библиотеки, в которые он может включить те компоненты библиотек Simulink, которые ему необходимы. Механизм построения таких библиотек реализован в Simulink.

Ниже описывается состав основной библиотеки Simulink и дается характеристика некоторых прикладных библиотек. Описание каждого компонента дополняется файлом-примером, иллюстрирующим его работу. Такие файлы размещены в папке, название которой совпадает с названием соответствующей библиотеки, а имя файла, как правило, очень близко к названию компонента. Для запуска файла-примера достаточно его открыть в Simulink и выполнить моделирование.

При описании компонентов наименования его параметров выделены жирным шрифтом. Описание наиболее важных параметров компонента дается при его описании. Те параметры, которые являются характерными для большого количества компонентов данного раздела библиотеки или всей библиотеки описываются при характеристике этого раздела или библиотеки. Поскольку некоторые компоненты включены сразу в несколько разделов библиотеки или в различные библиотеки, то полное описание такого компонента дается

только в одном разделе (обычно в том, в котором он встречается впервые, если следовать порядку расположения разделов библиотеки, принятому разработчиками Simulink), а в остальных разделах дается краткое описание и приводятся ссылки на указанный раздел.

Описание каждой библиотеки предваряется графическим изображением ее состава, которое является копией рабочего окна этой библиотеки в Simulink.

Наиболее общими параметрами, используемыми большинством компонентов, Simulink являются **Sample time** и **Interpret vector parameters as 1-D**.

Sample time – тактовый интервал модельного времени, через который рассчитываются сигналы на входе или выходе блока. Если **Sample time** = -1, то этот параметр наследуется от предыдущего блока. Если **Sample time** = 0, то он устанавливается равным минимальному шагу модельного времени, определяемого Simulink на основе установок в окне конфигурации параметров. Это значение характерно для моделирования непрерывных систем. Значение **Sample time** > 0 устанавливают при моделировании дискретных систем.

Interpret vector parameters as 1-D – флаг, устанавливаемый напротив соответствующего параметра позволяет интерпретировать этот параметр как вектор (одномерный массив).

3.1. Commonly Used Blocks - блоки общего применения

Библиотека блоков общего применения (Commonly Used Blocks) – первый раздел библиотеки в разделе Simulink. Он содержит блоки, показанные на рис. 3.1. Эти блоки наиболее часто используются при построении моделей в Simulink. Все компоненты этого раздела продублированы в специализированных разделах. Ниже (табл. 3.1) описана функция каждого блока включенного в эту библиотеку, а в папке Commonly Used Blocks содержатся файлы-примеры, иллюстрирующие их работу.

Commonly Used Blocks

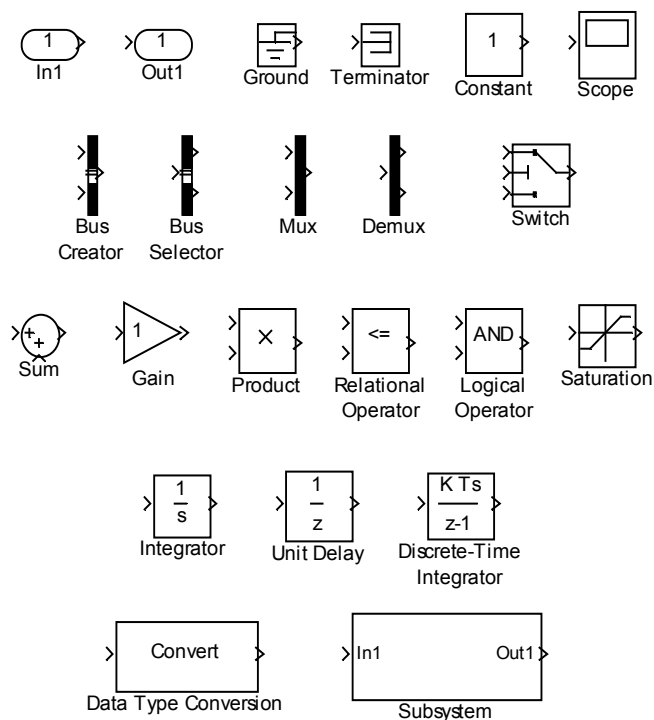

















Рис. 3.1

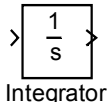
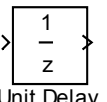
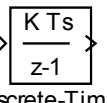
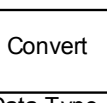
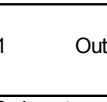
Таблица 3.1

Раздел Simulink|Commonly Used Blocks

Имя блока	Назначение ,краткое описание	Пример модели
In1 	Блок In1(Inport – входной порт) является входным портом подсистемы, а также служит для передачи в модель сигналов извне. Подробнее описан ниже (табл.3.8)	Inport_.mdl
Out1 	Блок Out1(Outport – выходной порт) является выходным портом подсистемы , а также служит для передачи сигналов из системы во внешние системы. Подробнее описан ниже (табл.3.9)	OutPort_.mdl
Ground 	Блок Ground (земля) можно использовать, чтобы "заземлить" входные порты блоков, не подключенные к другим блокам. Это блокирует вывод предупреждающих сообщений при моделировании. Блок Ground выдает нулевой сигнал на вход порта, к которому он подключен.	Ground_.mdl
Terminator	Блок Terminator можно использовать как "заглушку" на выходной порт любого блока, не подключенный к другим блокам. Это позволяет подавить вывод преду-	Term_.mdl

 Terminator	<p>преждающих сообщений о не использовании таких портов при моделировании.</p>	
<p>Constant</p>  Constant	<p>Constant block генерирует действительную или комплексную постоянную величину: скаляр, вектор (одномерный массив) или матрицу (двумерный массив), в зависимости от формы ввода генерируемой константы в поле Constant Value окна установки параметров. Подробнее описан ниже (табл.3.8)</p>	<p>Const_.mdl</p>
<p>Scope</p>  Scope	<p>Блок Scope (осциллограф) отображает форму сигнала как функции времени моделирования. Блок может иметь несколько входов и отображать сигналы на них в общем временном диапазоне (подробнее см. табл. 3.9).</p>	<p>Scope_.mdl</p>
<p>Bus Creator Bus Selector</p>  	<p>Блок Bus Creator используется для создания сигнальной шины, а Bus Selector - чтобы получить доступ к компонентам шины. Подробнее описаны ниже (табл.3.7)</p> <p>Свойства объектов типов шины позволяет изменять редактор шины (Bus Editor). Bus Editor вызывается через меню Tools или всплывающее меню.</p>	<p>Bus_Creator.mdl Bus_Selector.mdl</p>
<p>Mux Demux</p>  Mux Demux	<p>Блок Mux (мультиплексор) объединяет Number of inputs входных сигналов в единый выходной сигнал. Входной сигнал может быть скаляром, вектором, или матричным сигналом.</p> <p>Demux (демультиплексор) разделяет вектор входного сигнала на векторы сигналов меньшей размерности. Параметр Number of outputs позволяет определять число выходных портов и размерность каждого выходного порта.</p>	<p>Mux_.mdl DeMux_.mdl</p>
<p>Switch</p>  Switch	<p>Блок Switch (ключ) подключает к выходу первый или третий вход в зависимости от величины сигнала на втором входе. Первый и третий входы – входы данных, второй вход управляющий. Если условие Criteria for passing first input выполняется, то к выходу подключается первый вход, в противном случае – третий. Выражения "u2>=Threshold" и "u2>Threshold" означают соответственно "равенство или превышение" и "превышение" сигналом на управляющем входе порога Threshold, а выражение "u2~=0" – "u2 не равно нулю".</p>	<p>Switch_.mdl</p>
<p>Sum</p>	<p>Блок выполняет сложение или вычитание сигналов на своих входах. Он может складывать или вычитать скаляры, векторы или матрицы. Алгоритм работы блока задается параметром List of Signs: Символы плюс (+),</p>	<p>Sum_.mdl</p>

	<p>минус (-) и разделитель (!) указывает действия, которые необходимо выполнить над входными сигналами. Количество символов определяет количество входов. Например, комбинация "+-+" обозначает сумматор с тремя входами, причем сигналы на первом и третьем входах складываются, а сигнал на втором входе вычитается из суммы.</p>	
<p>Gain</p>  <p>Gain</p>	<p>Блок Gain (усилитель) умножает входной сигнал на константу Gain. Входной сигнал и константу могут быть скалярами, векторами или матрицами. Параметр Multiplication позволяет определять поэлементное Element-wise($K*u$) или матричное умножение. Для матричного умножения, этот параметр позволяет указывать порядок сомножителей Element-wise($K*u$) Matrix($u*K$). Если входной и выходной сигналы векторы - Matrix($K*u$)(u vector).</p>	<p>Gain_.mdl Gain_V.mdl</p>
<p>Product</p>  <p>Product</p>	<p>Блок Product выполняет умножение или деление чисел на его входах. Блок может использоваться для поэлементного или матричного умножения, в зависимости от параметра Multiplication. Специфицируя параметр Number of inputs (например: **/*) можно определять количество входов и задавать действия с числами на каждом из входов.</p>	<p>Product_.mdl</p>
<p>Relational Operator</p>  <p>Relational Operator</p>	<p>Блок Relational Operator сравнивает сигналы на своих двух входах в соответствии с критерием сравнения Relational Operator и выдает на выход логический сигнал TRUE (ИСТИНА), если выполняется условие, заданное указанным критерием.</p>	<p>Rel_Op.mdl</p>
<p>Logical Operator</p>  <p>Logical Operator</p>	<p>Блок Logical Operator выполняет логические операции над сигналами на его входах. Вид логической операции (Operator) и число входов блока (Number of input ports) задаются в окне Function Block Parameters. В блоке реализованы следующие виды логических операций: AND – И, OR – ИЛИ, NOT – НЕ, NAND – И-НЕ, NOR – ИЛИ-НЕ, XOR – исключающее ИЛИ (сумматор по модулю 2)</p>	<p>Logic_Op.mdl</p>
<p>Saturation</p>  <p>Saturation</p>	<p>Блок Saturation ограничивает верхний и нижний уровни сигнала, когда входной сигнал выходит за пределы диапазона определенного верхней (Upper limit) и нижней (Lower limit) границами.</p>	<p>Saturation_. mdl</p>
<p>Integrator</p>	<p>Блок Integrator выполняет интегрирование сигнала на</p>	

 <p>Integrator</p>	<p>своем входе. Можно использовать различные численные методы интегрирования, доступные в MatLab. Блок позволяет реализовать различные режимы работы интегратора: с внутренней и внешней установкой начального состояния Initial condition, с разными вариантами сброса External reset, с ограничением выходного сигнала Limit output сверху и снизу порогами Upper saturation limit и Lower saturation limit. Порт State позволяет избежать возникновения алгебраических циклов при интегрировании. Такие циклы возникают, если два или больше блоков связаны напрямую через петлю обратной связи.</p>	<p>Integr_.mdl</p>
<p>Unit Delay</p>  <p>Unit Delay</p>	<p>Блок Unit Delay задерживает сигнал на выходе на один такт дискретного сигнала. Этот блок является эквивалентом оператора дискретного времени z^{-1}. Он необходим для дискретизации сигнала по времени или при восстановлении квантованного сигнала.</p>	<p>Unit_Delay. mdl</p>
<p>Discrete-Time Integrator</p>  <p>Discrete-Time Integrator</p>	<p>Блок Discrete-Time Integrator выполняет интегрирование или накопление сигнала в дискретном времени. Этот блок является аналогом блока Integrator для непрерывного времени. Он включен также в раздел библиотеки Discrete Library. Он может использовать при вычислениях прямой и обратный методы Эйлера и метод трапеций.</p>	<p>Discr_Int.mdl</p>
<p>Data Type Conversion</p>  <p>Data Type Conversion</p>	<p>Блок Data Type Conversion преобразовывает входной сигнал любого типа данных Simulink в требуемый тип данных и масштабирует выбранным способом выходные данные. Входной и выходной сигналы могут быть действительными или комплексными. При использовании блока необходимо указать тип данных и/или масштаб для преобразования</p>	<p>Convert_.mdl</p>
<p>Subsystem</p>  <p>Subsystem</p>	<p>Блок Subsystem используется для представления части системы в виде подсистемы. С увеличением сложности модели можно повысить ее наглядность, группируя блоки в подсистемы. Библиотека Simulink содержит достаточно обширное множество средств построения подсистем с различными принципами построения (см. раздел Ports&Subsystems).</p>	<p>Ris_1_19.mdl Ris_1_20.mdl Sub_.mdl Trigger_.mdl ET_Sub.mdl Conf_Sub.mdl</p>

3.2. Continuous - линейные блоки непрерывного времени

Этот раздел библиотеки содержит блоки, используемые при моделировании динамических систем непрерывного времени.

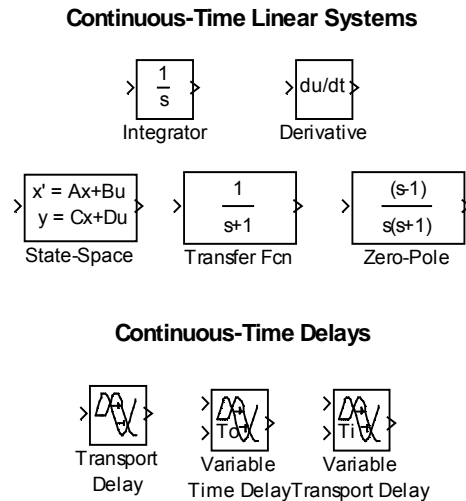


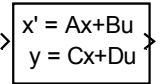
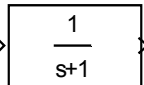
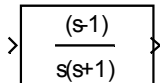




Рис. 3.2

Таблица 3.2

Раздел Simulink| Continuous

Имя блока	Назначение, краткое описание	Пример модели
Integrator  Integrator	Блок Integrator выполняет интегрирование сигнала на своем входе. Можно использовать различные численные методы интегрирования, доступные в MatLab. Блок позволяет реализовать различные режимы работы интегратора: с внутренней и внешней установкой начального состояния Initial condition , с разными вариантами сброса External reset , с ограничением выходного сигнала Limit output сверху и снизу порогами Upper saturation limit и Lower saturation limit . Порт State позволяет избежать возникновения алгебраических циклов при интегрировании. Такие циклы возникают, если два или больше блоков связаны напрямую через петлю обратной связи.	Integr_.mdl

Derivative  Derivative	Блок Derivative вычисляет приближенное значение производной входного сигнала. Блок позволяет выполнять операции над одним входным сигналом. Начальное значение сигнала на выходе блока нулевое. Чтобы линеаризовать нелинейные модели, в которых возможны скачки производной сигнала, можно использовать функцию linmod MatLab, либо использовать блок Switched derivative for linearization из раздела Simulink Extracts Linearization.	Deriv_.mdl
State-Space  State-Space	Блок State-Space реализует систему, определенную уравнениями состояния $\dot{s} = As + Bx$ $y = Cs + Dx$, где s, x, y - векторы-столбцы; A, B, C и D матрицы. x - вектор переменных состояния системы, x и y - входной и выходной сигналы. Параметры A и B – матрицы $n \times n$ и $n \times m$; C и D – матрицы $r \times n$ и $r \times m$; n – размерность вектора состояний, m – число входов, а r – число выходов системы	St_Space.mdl State4.mdl
Transfer Fcn  Transfer Fcn	Блок Transfer Fcn реализует линейную систему с операторным коэффициентом передачи (передаточной функцией) $K(s) = \frac{Y(s)}{X(s)}$, где $Y(s)$ и $X(s)$ – лапласовские изображения сигналов на выходе и входе системы; Numerator coefficient и Denominator coefficient - векторы коэффициентов полинома числителя и знаменателя.	Trans_Fun.mdl
Zero-Pole  Zero-Pole	Блок Zero-Pole реализует линейную систему, представленную набором нулей и полюсов ее передаточной функции $K(s)$. Параметры Zeros и Poles – векторы нулей и полюсов передаточной функции, Gain – коэффициент передачи для нулевой частоты.	Zero-Pole.mdl
Transport Delay  Transport Delay	Блок Transport Delay задерживает входной сигнал на время Time delay . Параметр Time delay (время задержки) не должен быть отрицательным числом. Точность работы блока тем выше, чем в большей степени задержка превышает интервал дискретизации сигнала при моделировании.	Delay_.mdl
Variable Time Delay block Variable Transport	Блоки Variable Time Delay block и Variable Transport Delay block представлены как два блока в библиотеке Simulink. Тем не менее, это один и тот же блок с разными типами параметра задержки, определяемыми установкой параметра Select delay type . Блок Variable Transport Delay моделирует задержку, когда скорость распространения сигнала в	Delay_.mdl

Delay block 	канале зависит от времени (подробнее см. Help на соответствующий блок). Если $u(t)$ - входной сигнал, $td(t)$ задержка, а $y(t)$ выходной сигнал, тогда $y(t) = u(t - td(t))$. Блоки имеют два входа, на один из которых подается задерживаемый сигнал, на другой сигнал управления задержкой. Maximum delay – максимальное время задержки (если задержка превысит это значение, блок ограничит ее величину).	
---	--	--

3.3. Discontininuous - нелинейные блоки

Основным параметром нелинейного устройства является его передаточная функция – зависимость выходного сигнала от входного $y(u)$. Раздел Nonlinear основной библиотеки Simulink содержит нелинейные блоки с наиболее распространенными передаточными функциями, используемые при моделировании динамических систем (рис. 3.3):

идеальные ограничители с постоянными и динамически изменяемыми порогами ограничения.

блоки нелинейности, моделирующие зону нечувствительности (постоянной и динамически изменяющейся),

блоки нелинейности, моделирующие петли гистерезиса,

блок нелинейности, моделирующий характеристику квантователя сигналов.

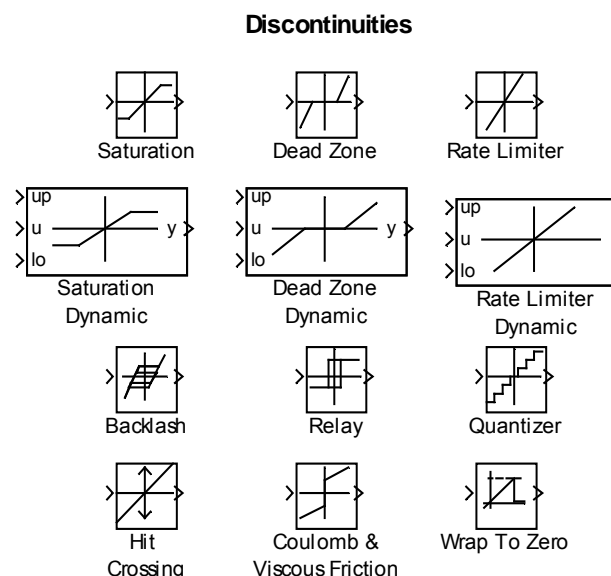

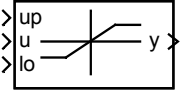

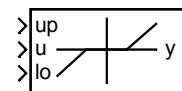

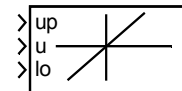








Рис. 3.3

Таблица 3.3

Раздел Simulink|Discontinious

Имя блока	Назначение ,краткое описание	Пример модели
Saturation  Saturation Saturation Dynamic  Saturation Dynamic	<p>Блоки Saturation представляет собой идеальный ограничитель, сигнал на выходе которого равен входному сигналу до тех пор, пока не достигнет порога ограничения: верхнего Upper limit или нижнего Lower limit. Флаг Treat as gain when linearizing определяет режим работы блока, когда входной сигнал находится между Lower limit и Upper limit. Когда флаг включен – режим линейный. Блок Saturation Dynamic отличается от блока Saturation тем, что его пороги ограничения могут управляться внешним сигналом</p>	Saturation_.mdl
Dead Zone  Dead Zone Dead Zone Dynamic  Dead Zone Dynamic	<p>Блок Dead Zone моделирует нелинейные элементы с зоной нечувствительности, у которых зависимость выходного сигнала от входного линейна везде за исключением некоторой “мертвой зоны” – зоны нечувствительности, задаваемой границами Start of dead zone и End of dead zone. Внутри этой зоны выходной сигнал равен нулю</p> <p>Блок Dead Zone Dynamic отличается от блока Dead Zone тем, что границы зоны нечувствительности могут управляться внешним сигналом</p>	Dead_Zone.mdl
Rate Limiter  Rate Limiter Rate Limiter Dynamic  Rate Limiter Dynamic	<p>Блок Rate Limiter ограничивает скорость изменения выходного сигнала пределами Rising slew rate и Falling slew rate. Если скорость изменения входного сигнала находится в этих границах, то входной и выходной сигналы совпадают.</p> <p>Когда скорость изменения входного сигнала превысит Rising slew rate, выходной сигнал y_i вычисляется по формуле</p> $y_i = dt_m * \text{Rising slew rate} + y_{i-1},$ <p>где i – номер текущего шага моделирования, dt_m – приращение времени на текущем шаге модельного времени. Если скорость изменения входного сигнала окажется меньше Falling slew rate</p> $y_i = dt_m * \text{Falling slew rate} + y_{i-1}.$ <p>Блок Rate Limiter Dynamic отличается от блока Rate</p>	Rate_Lim.mdl Rate_Lim_0.mdl

	Лimiter тем, что пределы скорости могут изменяться внешним сигналом	
Backlash  Backlash	Блок Backlash имитирует эффект возникновения люфта - передаточную характеристику гистерезисного типа. Параметр Deaband width (ширина диапазона) определяет ширину петли гистерезиса передаточной характеристики, а параметр Initial output – начальный уровень сигнала на выходе. Сигнал на выходе будет равен Initial output, пока при возрастании не достигнет значения $U_{ex} + (Deaband\ width)/2$, после чего изменяться не будет. При спаде сигнал перестает меняться, достигнув границы $U_{ex} - (Deaband\ width)/2$.	Backlash.mdl
Relay  Relay	Блок Relay имеет разрывную передаточную функцию релейного типа (с гистерезисом или без него). Выходной сигнал блока принимает значение Output when on при достижении входным сигналом порогового уровня Switch on point . При уменьшении входного сигнала до порога Switch off point на выходе устанавливается уровень Output when off .	Relay.mdl
Quantizer  Quantizer	Блок Quantizer служит для квантования сигналов по уровню с шагом Quantization interval . Сигналы квантуются по уровню и превращаются в ступенчатые сигналы.	Quant.mdl
Hit Crossing  Hit Crossing	Блок Hit Crossing фиксирует прохождение сигналом заданного уровня Hit crossing offset и при каждом его пересечении в направлении Hit crossing directions (either – любое, rising – нарастание, failing – спад) вырабатывает короткий импульс единичной амплитуды.	Hit Cross.mdl
Coulombic and Viscous Friction  Coulomb & Viscous Friction	Блок Coulombic and Viscous Friction служит для моделирования фрикционных эффектов сухого и вязкого трения. Параметры блока: Coulomb friction value (Offset) – список смещений при фрикционных эффектах и Coefficient of viscous friction (Gain) – коэффициент вязкого трения (коэффициент передачи для приращений выходного сигнала).	Vis_Frict.mdl
Wrap To Zero  Wrap To Zero	Блок Wrap To Zero устанавливает выходной сигнал в нуль, когда входной сигнал превысит порог Threshold и повторяет входной сигнал, если он не превосходит Threshold .	Wrap_To_Z.mdl

3.4. Discrete -дискретные блоки

Дискретные блоки служат для построения моделей дискретных устройств и систем двух типов: с дискретным временем и с дискретными состояниями. Эти блоки включают в себя модели типичных цифровых устройств: устройство цифровой задержки, дискретно-временной интегратор, дискретный фильтр и т. п. Сюда же включены блок весового суммирования отсчетов сигнала, блок памяти и два блока экстраполяции сигнала по его дискретным значениям. Раздел библиотеки Discrete показан на рис. 3.4.

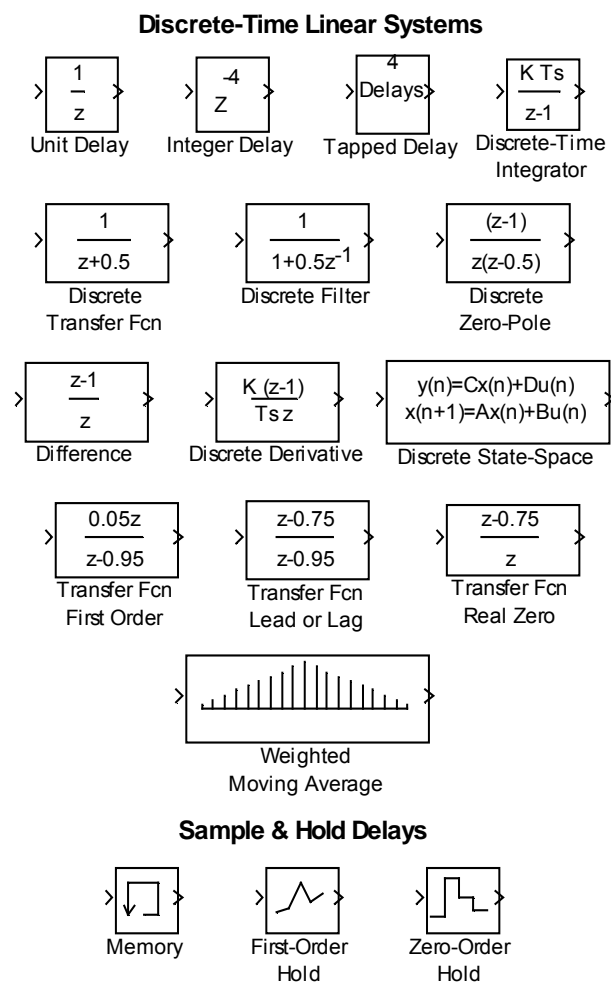
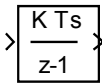
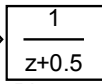
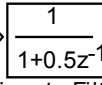
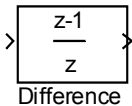
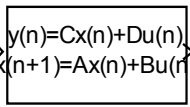
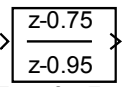


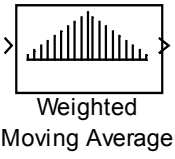
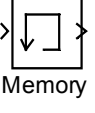
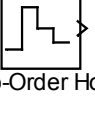

Рис. 3.4

Таблица 3.4

Раздел Simulink| Discrete

Имя блока	Назначение, краткое описание	Пример модели
Unit Delay Integer Delay Tapped Delay	<p>Блок Unit Delay задерживает входной сигнал на один такт Simple Time. Параметр Initial condition – начальное значение выходного сигнала</p> <p>Блок Integer Delay задерживает входной сигнал на Number of delays тактов Simple Time</p> <p>Блок Tapped Delay формирует шину выходных сигналов размером Number of delays, каждый из сигналов, которой задержан относительно предыдущего на один такт Simple Time</p>	<p>Discr_Delay.mdl Tapped_Delay.mdl</p>
Discrete-Time Integrator  Discrete-Time Integrator	<p>Discrete-Time Integrator интегратор сигнала в дискретном времени. Может использоваться для управления логикой работы модели, например для остановки процесса моделирования по заданному значению интеграла времени.</p> <p>Порт State позволяет избежать возникновения алгебраических циклов при интегрировании. Такие циклы возникают, если два или больше блоков связаны напрямую через петлю обратной связи.</p> <p>Integration method – метод численного интегрирования; External reset – сброс внешним сигналом; Initial condition – начальное значение выходного сигнала; Limit output – ограничения на изменение выходного сигнала сверху (Upper) и снизу (Lower).</p>	Discr_Int.mdl
Discrete Transfer Fcn  Discrete Transfer Fcn Discrete Filter  Discrete Filter Discrete Zero Pole	<p>Блок Discrete Transfer Fcn предназначен для моделирования дискретных систем с передаточной функцией, заданной коэффициентами полиномов числителя Numerator coefficient и знаменателя Denominator coefficient относительно переменной z.</p> <p>Блок Discrete Filter предназначен для моделирования дискретных фильтров с передаточной функцией, заданной коэффициентами полиномов числителя Numerator coefficient и знаменателя Denominator coefficient относительно переменной z^{-1}.</p> <p>Блок Discrete Zero Pole предназначен для моделирования дискретных систем с передаточной функцией заданной нулями Zeros и полюсами Poles передаточной функции и коэффициентом передачи Gain.</p>	<p>Diskr_Tr_Fcn.mdl Diskr_Filter.mdl Diskr_zp.mdl</p>

<p>Difference</p>  <p>Difference</p>	<p>Блок Difference вычисляет разность y_n между текущим и предыдущим отсчетами входного сигнала: $y_n = x_n - x_{n-1}$.</p> <p>Значение нулевого отсчета задается параметром Initial condition for previous output.</p> <p>Блок обрабатывает сигналы в любом формате, поддерживаемом Simulink, включая формат с фиксированной точкой.</p>	<p>Discr_Deriv.mdl</p>
<p>Discrete Derivative</p>	<p>Блок Discrete Derivative вычисляет производную y_n сигнала x_n для дискретного времени</p> $y_n = \text{Gain value} \cdot (x_n - x_{n-1}) / \text{Simple Time}$	<p>Discr_Deriv.mdl</p>
<p>Discrete State-Space</p>  <p>Discrete State-Space</p>	<p>Блок Discrete State-Space реализует линейную дискретную систему, которая описывается системой уравнений</p> $\begin{aligned} s_{n+1} &= A \cdot s_n + B \cdot x_n \\ y_n &= C \cdot s_n + D \cdot x_n \end{aligned}$ <p>где x и y – входной и выходной сигналы, s – вектор состояний системы, A, B, C и D – матрицы, описывающие моделируемую систему в переменных состояния s. Размерность вектора x определяется количеством столбцов в матрицах B и D, а вектора y – количеством строк в матрицах C и D.</p>	<p>Diskr_SS.mdl</p>
<p>Transfer Fcn First Order</p> <p>Transfer Fcn Lead or Lag</p>  <p>Transfer Fcn Lead or Lag</p> <p>Transfer Fcn Real Zero</p>	<p>Блок Transfer Fcn First Order реализует дискретную передаточную функцию первого порядка с полюсом задаваемым в поле Pole (in Z plane) и единичным коэффициентом передачи на нулевой частоте.</p> <p>Блок Transfer Fcn Lead or Lag реализует в дискретном времени компенсацию опережения или отставания фазы входного сигнала Коэффициент передачи блока на нулевой частоте определяется выражением $(1-\text{Zero})/(1-\text{Pole})$, где Zero - нуль и Pole - полюс передаточной функции блока. На остальных частотах он равен единице.</p> <p>Блок Transfer Fcn Real Zero реализует функцию передачи дискретного времени с одним действительным нулем и отсутствием эффективно влияющих полюсов</p>	<p>Discr_Tr_Fcn_Cl.mdl</p>
<p>Weighted Moving Average</p>	<p>Блок Weighted Moving Average вычисляет взвешенное текущее среднее значение сигнала. Он запоминает N последних отсчетов входного сигнала и умножает каждый из них на некоторое число - весовой множитель, определенной параметром Weights и группирует их в вектор. Блок поддерживает режимы один вход/один выход (SISO) и один вход/много выходов (SIMO). Для режима</p>	<p>Weight_Aver.mdl</p>

	<p>SISO, параметр Weights задается как вектор-столбец, для SIMO как матрица, каждый столбец которой соответствует отдельному выходу.</p>	
<p>Memory</p> 	<p>Блок Memory (Память) запоминает входной сигнал и смещает его на один такт времени. Если установлен флаг Inherit sample time, то тактовый интервал устанавливается равным интервалу Sample time предшествующего блока. При сброшенном флаге длительность такта равна 0,1 модельного времени.</p>	<p>Memory_.mdl</p>
<p>Zero-Order Hold</p>  <p>First-Order Hold</p> 	<p>Блок Zero-Order Hold - экстраполятор нулевого порядка – он устанавливает выходной сигнал, равный входному в предыдущей тактовой точке, и не изменяет его на интервале Sample time. Входной и выходной сигналы могут быть скалярами или векторами.</p> <p>Блок First-Order Hold - линейный экстраполятор входного сигнала: на интервале Sample time сигнал экстраполируется его первой производной в предыдущей тактовой точке. Он редко применяется на практике и включен в библиотеку преимущественно для исследовательских целей.</p>	<p>Hold_.mdl</p>

3.5. Logic and Bit Operations - логические и битовые операции

Библиотека логических и битовых операций Logic and Bit Operations (рис. 3.5) содержит три подраздела: Logic Operations – блоки выполнения логических операций, Bit Operations – блоки выполнения битовых операций и Edge Detection – блоки анализа граничных ситуаций.

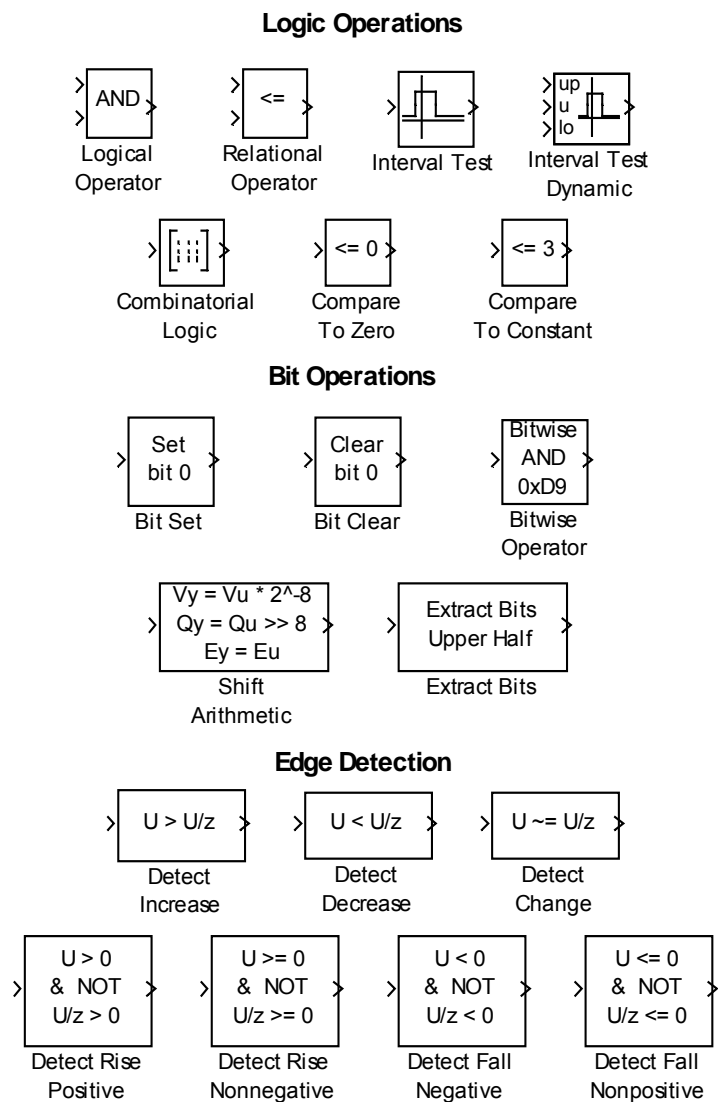


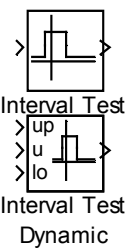
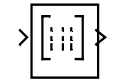
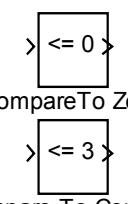


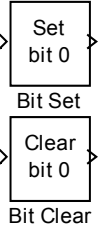
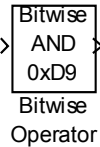
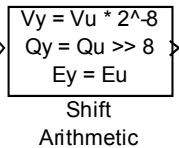
Рис. 3.5

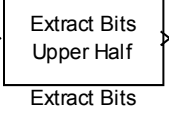
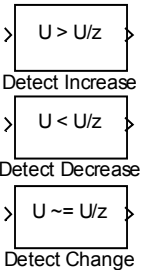
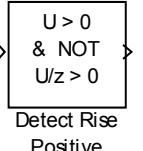
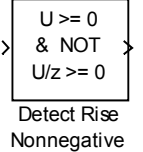
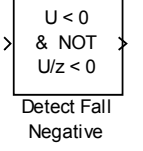
Таблица 3.5

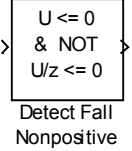
Раздел Simulink| Logic and Bit Operations

Имя блока	Назначение, краткое описание	Пример модели
Logical Operator  <p>Logical Operator</p>	<p>Блок Logical Operator выполняет заданную логическую операцию над сигналами на его входах. Входной сигнал считается TRUE (1), если он отличен от нуля и FALSE (0), если равен нулю.</p> <p>Выполняемая блоком логическая операция выбирается из раскрывающегося списка Operator: AND – И, OR – ИЛИ, NOT – НЕ, NAND – И-НЕ, NOR – ИЛИ-НЕ, XOR – исключающее ИЛИ (сумматор по модулю 2).</p>	Logic_Op.mdl

	Количество входов задается параметром Number of input ports . Параметром Icon shape можно выбирать форму изображения блока в модели	
Relational Operator  Relational Operator	Блок Relational Operator сравнивает сигналы на своих двух входах в соответствии с критерием сравнения Relational Operator , заданным в окне Function Block Parameters, и выдает на выход логический сигнал TRUE (ИСТИНА), если выполняется условие, заданное критерием сравнения (== – равно, ~=.– не равно, > – больше, < – меньше, >= – не менее, >= – не более).	Rel_Op.mdl
Interval Test (Interval Test Dynamic)  Interval Test Interval Test Dynamic	Блок Interval Test выдает TRUE (1), если входной сигнал находится внутри интервала Lower limit ÷ Upper limit . Этот интервал может быть открытым, закрытым справа или слева в зависимости от установки флагов Interval closed on left и Interval closed on right . Блок Interval Test Dynamic отличается от блока Interval Test наличием управляющих входов, на которые подаются сигналы, изменяющие Lower limit и Upper limit в процессе моделирования.	Interv_Test.mdl
Combinatorial Logic  Combinatorial Logic	Блок Combinatorial Logic (комбинационная логика) реализует логическое устройство задаваемое таблицей истинности. Его можно использовать вместе с блоками Memory, чтобы моделировать устройства типа конечных автоматов или триггеров Логика работы блока определяется матрицей Truth table , представляющей таблицу истинности. Между количеством входов блока и количеством строк матрицы должно выполняться соотношение <i>Количество строк = 2 ^ (Количество входов).</i> Количество столбцов в матрице равно сумме числа входов и выходов.	Comb_Logic.mdl
Compare To Zero (Compare To Constant)  Compare To Zero Compare To Constant	Блок Compare To Zero сравнивает входной сигнал с нулем. Логика сравнения входного сигнала с нулем определяется параметром Operator (==, ~=, >, <, >=, <=). Входные сигналы могут быть любых типов, поддерживаемых Simulink, выходные - uint8 или Boolean. Блок Compare To Constant отличается от блока Compare To Zero тем, что входной сигнал сравнивается не с нулем, а с константой Constant value .	Compare_To.mdl
Bit Set	Блок Bit Set устанавливает определенные биты компо-	Bit_Set.mdl

<p>(Bit Clear)</p> 	<p>нент входного вектора в состояние '1'. Например, задав параметр Index of bit равным [3 0 1 2], можно установить в '1' третий, нулевой, первый и второй разряды первого, второго, третьего и четвертого компонент вектора входного сигнала.</p> <p>Блок Bit Clear отличается от блока Bit Set только тем, что соответствующие биты устанавливаются в состояние '0'.</p>	
<p>Bitwise Operator</p> 	<p>Блок Bitwise Operator выполняет требуемые поразрядные операции над своими операндами. В отличие от логических действий, выполняемых блоком Logical Operator, Bitwise Operator рассматривают операнды как вектор битов, а не единое число. Вид требуемой поразрядной операции определяется параметром Operator. С помощью флага Use bit mask можно получить доступ к параметру Bit Mask (битовая маска) и указать в нем те биты, над которыми необходимо выполнить заданные параметром Operator логические операции.</p>	<p>Bit_wise.mdl</p>
<p>Shift Arithmetic</p> 	<p>Блок Shift Arithmetic (арифметический сдвиг) можно использовать для сдвига битов или двоичной точки (или и того и другого одновременно) в числе на его входе. Number of bits to shift right – число бит, на которое целое число сдвигается вправо (если число отрицательное – влево). Number of places by which binary point shifts right - число бит, на которое точка в числе формата sfix(8) или sfix(16) сдвигается вправо (если число отрицательное – влево). В результате сдвига точки в двоичном числе 11001.011=-6.625 типа sfix(8) на два разряда вправо и два разряда влево получим: 1100101.1 = -26.5 и 110.01011=-1. 65625.</p> <p>Блок выполняет арифметические битовые сдвиги в числах со знаками. Следовательно, самый старший двоичный разряд перемещается после каждого сдвига битов. Сдвигая число 11001.011=-6.625 типа sfix(8) на два разряда вправо и два разряда влево получим: 11110.010= -1.75; 00101.100= 5.5 .</p>	<p>Shift_Arith.mdl</p>

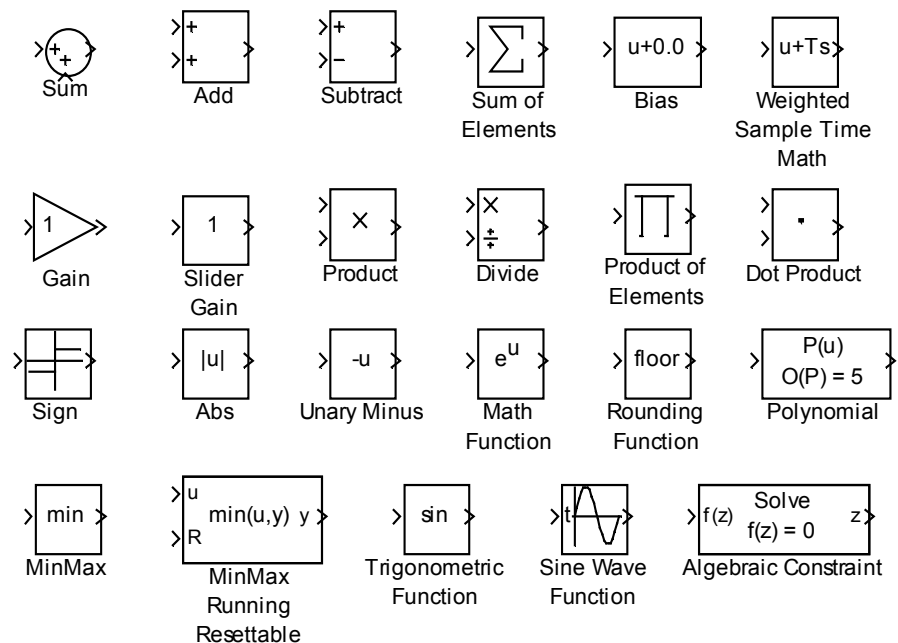
<p>Extract Bits</p> 	<p>Блок Extract Bits позволяет выделить требуемую последовательность бит из входного сигнала. Выделяемые биты задаются параметром Bits to extract: Upper half или Lower half – половина старших или младших бит; Range starting with most significant bit или Range ending with least significant bit – Number of bits с начала или с конца; Range of bits – биты в диапазоне $[k_{нач} \ k_{кон}]$, задаваемом параметром Bit indices.</p>	<p>Extract_Bits.mdl</p>
<p>Detect Increase (Detect Decrease Detect Change)</p> 	<p>Блок Detect Increase определяет: увеличился ли входной сигнал в сравнении с предыдущим отсчетом и устанавливает выходной сигнал равным True (1), если это произошло. В противном случае выходной сигнал равен False (0).</p> <p>Блок Detect Increase отличается от блока Detect Increase тем, что обнаруживает уменьшение входного сигнала в сравнении с предыдущим отсчетом, а блок Detect Change – изменение входного сигнала.</p>	<p>Detect_.mdl</p>
<p>Detect Rise Positive</p>  <p>Detect Rise Nonnegative</p>  <p>Detect Fall Negative</p>  <p>Detect Fall Nonpositive</p>	<p>Блок Detect Rise Positive устанавливает выходной сигнал равным True (1), если входной сигнал в текущем отсчете стал больше нуля, а в предыдущем был меньше чем нуль. В противном случае выходной сигнал равен False (0).</p> <p>Блок Detect Rise Nonnegative устанавливает выходной сигнал равным True (1), если входной сигнал в текущем отсчете оказался не меньше нуля, а в предыдущем был меньше чем нуль.</p> <p>Блок Detect Fall Negative устанавливает выходной сигнал равным True (1), если входной сигнал в текущем отсчете стал меньше нуля, а в предыдущем был не меньше нуля.</p>	<p>Detect_Rise.mdl</p>

	Блок Detect Fall Nonpositive устанавливает выходной сигнал равным True (1), если входной сигнал в текущем отсчете стал меньше нуля, а в предыдущем был больше нуля.	
---	---	--

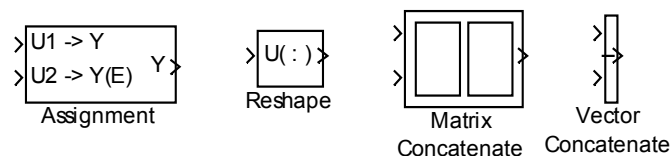
3.6. Math Operations - математические операции

Библиотека математических операций Math Operations содержит три подраздела: Math Operations – математические операции, Vector/Matrix Operations – операции над векторами и матрицами и Complex Vector Conversions – преобразования комплексных векторов.

Math Operations



Vector/Matrix Operations



Complex Vector Conversions

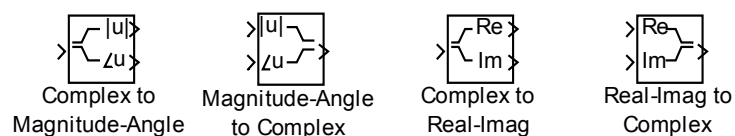
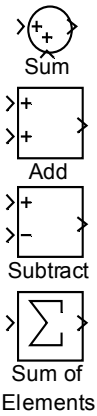



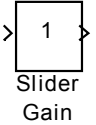
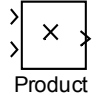
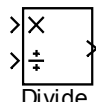
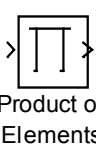
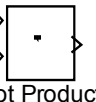
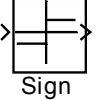
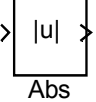

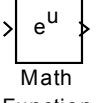

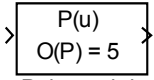

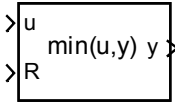




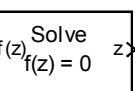
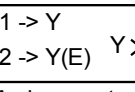

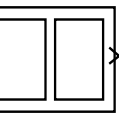
Рис. 3.6





Раздел Simulink| Math Operations

Имя блока	Назначение, краткое описание	Пример модели
Sum (Add, Subtract, Sum of Elements) 	<p>Эти блоки, по сути, являются одним и тем же блоком с разными названиями, графическими изображениями и соответствующими пред установками параметров. Они выполняют сложение или вычитание сигналов на своих входах, которые могут быть скалярами, векторами или матрицами.</p> <p>Алгоритм работы блоков задается параметром List of Signs: Символы плюс (+), минус (-) и разделитель () указывает действия, которые необходимо выполнить над входными сигналами. Количество символов определяет количество входов. Например, комбинация "+ - +" обозначает сумматор с тремя входами, причем сигналы на первом и третьем входах складываются, а сигнал на втором входе вычитается из суммы. Если List of Signs содержит только один символ, блок выполняет поэлементное суммирование (вычитание) компонентов входного вектора.</p>	Sum_.mdl
Bias 	<p>Блок Bias (смещение) добавляет константу Bias ко входному сигналу: $y = x + Bias$. Блок поддерживает все типы данных Simulink, за исключением Boolean (логического).</p>	Bias_.mdl
Weighted Sample Time Math 	<p>Блок Weighted Sample Time Math суммирует, вычитает, умножает, или делит входной сигнал в соответствии с весом Weight value. Параметр Operation определяет выполняемое математическое действие. Если параметр Operation выбрать равным T_s или $1/T_s$, то выходной сигнал блока соответственно будет равен Weight value или $1/Weight value$.</p>	Weigt_Samp.mdl
Gain 	<p>Блок Gain умножает входной сигнал на постоянный коэффициент Gain. Способ выполнения операции умножения определяется параметром Multiplication: Element-wise($K*u$) – поэлементный; Matrix($K*u$) и Matrix($u*K$) – матричный, когда K является левосторонним и правосторонним операндом; Matrix($K*u$)(u vector) – правостороннее матричное умножение, когда входной и выходной сигналы векторы, длина которых определяется размерностью матрицы усиления.</p>	Gain_.mdl Gain_V.mdl

Slider Gain 	Блок Slider Gain позволяет перемещением движка изменять в процессе моделирования скалярный коэффициент усиления в пределах Low – High .	Slider_Gain.mdl
Product (Divide, Product of Elements)   	<p>Блок Product выполняет умножение или деление чисел на его входах. Блок может использоваться для поэлементного или матричного умножения, в зависимости от параметра Multiplication. Специфицируя соответствующим образом параметр Number of inputs (например: **/* – сигналы с первого, второго и четвертого входов перемножаются и делятся на сигнал с третьего входа) можно определять количество входов и задавать действия с числами на каждом из входов.</p> <p>Блоки Divide и Product of Elements это блоки Product, в которых соответствующим образом специфицированы параметры Number of inputs и Multiplication. Блок Product of Elements выполняет поэлементное перемножение (деление) компонент вектора на его входе.</p>	Product_.mdl
Dot Product 	Блок Dot Product вычисляет скалярное произведение y двух векторов \vec{x}_1 и \vec{x}_2 : $y = \sum_{k=1}^n x_{1k}^* \cdot x_{2k}$. Если оба входных вектора действительные, то выходной сигнал тоже действительный, если хотя бы один комплексный – выходной сигнал комплексный.	Dot_.mdl
Sign 	Блок Sign определяет знак входного сигнала в соответствии с алгоритмом: выходной сигнал $y = 1$ когда входной сигнал $x > 0$, $y = 0$ при $x = 0$ и $y = -1$ при $x < 0$	Sign_.mdl
Abs 	Блок Abs вычисляет абсолютное значение действительного числа или модуль комплексного числа	Abs_.mdl
Unary Minus 	Блок Unary Minus инвертирует входной сигнал. Он не может работать с типами данных, для которых не определено понятие “знак”.	Unary_Min.mdl
Math Function 	Блок Math Function выполняет вычисление математической функции, задаваемой параметром Function : exp – вычисление экспоненты; log – вычисление натурального логарифма; 10^u – вычисление степени 10; log10 – вычисление десятичного логарифма;	Math_Fun.mdl

	<p> magnitude^2 – вычисление модуля; square – вычисление квадрата ; sqrt – вычисление квадратного корня; pow – возведение в степень conj – вычисление комплексно-сопряженного числа; reciprocal – вычисление обратной величины ($1/x$); hypot – вычисление корня квадратного из суммы квадратов; rem – вычисление остатка* от деления первого числа на второе; mod – вычисление остатка* от деления первого числа на второе; transpose – транспонирование матрицы; hermitian – вычисление эрмитовой матрицы. </p> <p>* Функции rem и mod отличаются способом округления остатка от деления: rem округляет остаток в направлении нуля, а mod – минус бесконечности.</p>	
<p>Rounding Function</p>  <p>Rounding Function</p>	<p>Блок Rounding Function выполняет операцию округления числа. Способ округления Function выбирается из списка: floor – до ближайшего целого в направлении $-\infty$; ceil – до ближайшего целого в направлении $+\infty$; round – до ближайшего целого; fix – до ближайшего целого в направлении нуля.</p>	Round_.mdl
<p>Polynomial</p>  <p>Polynomial</p>	<p>Блок Polynomial вычисляет значение полинома, заданного вектором коэффициентов Polynomial coefficients</p>	Polynom_.mdl
<p>MinMax</p>  <p>MinMax</p>	<p>Блок MinMax определяет максимальное или минимальное (Function) значение из всех Number of input ports сигналов, поступающих на его входы</p>	Min_Max.mdl
<p>MinMax Running Resettable</p>  <p>MinMax Running Resettable</p>	<p>Блок MinMax Running Resettable определяет максимальное или минимальное (Function) всех компонент вектора входного сигнала. Блок инициализируется положительным перепадом сигнала по входу R. Если последующее значение становится меньше (при анализе на max) или больше (при анализе на min), выходной сигнал остается равным значению в момент инициализации блок по входу R. Сигнал на выходе блок можно устанавливать равным величине задаваемой параметром Initial condition, если на вход R подать логическую единицу True (1).</p>	Min_Max.mdl

Trigonometric Function  Trigonometric Function	Блок Trigonometric Function выполняет вычисление тригонометрических функции, определяемой параметром Function : <i>sin</i> , <i>cos</i> , <i>tan</i> , <i>asin</i> , <i>acos</i> , <i>atan</i> , <i>atan2</i> , <i>sinh</i> , <i>cosh</i> , <i>tanh</i> , <i>asinh</i> , <i>acosh</i> и <i>atanh</i> . Функция <i>atan2</i> – арктангес отношения чисел на первом и втором входах.	Trig_Fun.mdl
Sine Wave  Sine Wave Function	Блок Sine Wave – генератор гармонического сигнала с параметрами Amplitude – амплитуда, Frequency – частота (рад/с), Phase – фаза (рад), Bias – смещение (постоянная составляющая). Параметр Time определяет, какое время используется для вычисления отсчетов выходного сигнала: Use external source - внешний источник; Use simulation time – внутреннее время.	Sine_Wave.mdl
Algebraic Constraint  Algebraic Constraint	Блок Algebraic Constraint выполняет поиск корней алгебраических уравнений, при этом выходной сигнал должен быть прямо или опосредованно связан с входным сигналом. Блок может использоваться для решения нелинейных матричных уравнений. Параметр Initial guess – приближенное значение корня – указывает значение аргумента, в окрестности которого ищется корень	Alg_Constr.mdl
Assignment  Assignment	Блок Assignment присваивает требуемые значения заданным элементам вектора или матрицы. Ко входу U1 подключается источник исходного сигнала, ко входу U2 – источник сигнала, который должен заменить элементы исходного сигнала. Индексы заменяемых элементов указываются в окне установки параметров блока. Для матрицы: Rows – строка; Columns – колонка. Для вектора: Elements – номер элемента	Assign_.mdl
Reshape  Reshape	Блок Reshape изменяет размерность сигнала в соответствии со способом, указанным в поле Output dimensionality . Например, можно преобразовать 2N-элементный вектор в матрицу N×N или наоборот, одномерный массив в вектор-строку или вектор-столбец и т.д.	Reshape_.mdl
Matrix Concatenate (Vector Concatenate)  Matrix Concatenate	Блок Matrix Concatenate (Vector Concatenate) объединяет две или более матриц (векторов) на своих входах в одну матрицу (вектор). Количество объединяемых матриц (векторов) задается параметром Number of inputs , способ объединения выбирается из списка Mode : Vector concatenation – объединение векторов; Horizontal matrix concatenation – объединение матриц по строкам; Vertical matrix concatenation – объединение матриц по столбцам.	
Complex to Magnitude-	Блок Complex to Magnitude-Angle вычисляет модуль и (или) аргумент комплексного числа. Выходной сигнал вы-	

<p>Angle</p>  <p>Complex to Magnitude-Angle</p>	<p>бирается из списка Output: Magnitude and angle – модуль и аргумент; Magnitude – модуль; Angle – аргумент (угол, рад). Входной сигнал блока может быть скалярным, векторным или матричным сигналом.</p>	
<p>Magnitude-Angle to Complex</p>  <p>Magnitude-Angle to Complex</p>	<p>Блок Magnitude-Angle to Complex вычисляет комплексное число по его модулю и (или) аргументу. Входной сигнал выбирается из списка Input: Magnitude and angle – модуль и аргумент; Angle – аргумент (угол, рад); Magnitude – модуль. Входной сигнал блока может быть скалярным, векторным или матричным сигналом.</p>	
<p>Complex to Real-Imag</p>  <p>Complex to Real-Imag</p>	<p>Блок Complex to Real-Imag вычисляет действительную и (или) мнимую части комплексного числа. Выходной сигнал выбирается из списка Output: Real – действительная часть; Imag – мнимая часть; Real and imag - действительная и мнимая части. Входной сигнал блока может быть скалярным, векторным или матричным сигналом.</p>	
<p>Real-Imag to Complex</p>  <p>Real-Imag to Complex</p>	<p>Блок Real-Imag to Complex вычисляет комплексное число по его действительной и (или) мнимой части. Входной сигнал выбирается из списка Output: Real – действительная часть; Imag – мнимая часть; Real and imag - действительная и мнимая части. Входной сигнал блока может быть скалярным, векторным или матричным сигналом.</p>	

3.7. Signal Routing - блоки маршрутизации сигналов

Библиотека блоков маршрутизации сигналов Signal Routing содержит два подраздела: Signal Routing – маршрутизация сигналов и Signal Storage & Access – запоминание, хранение и чтение сигналов из памяти. Она содержит средства для коммутации сигналов, их объединения и разделения

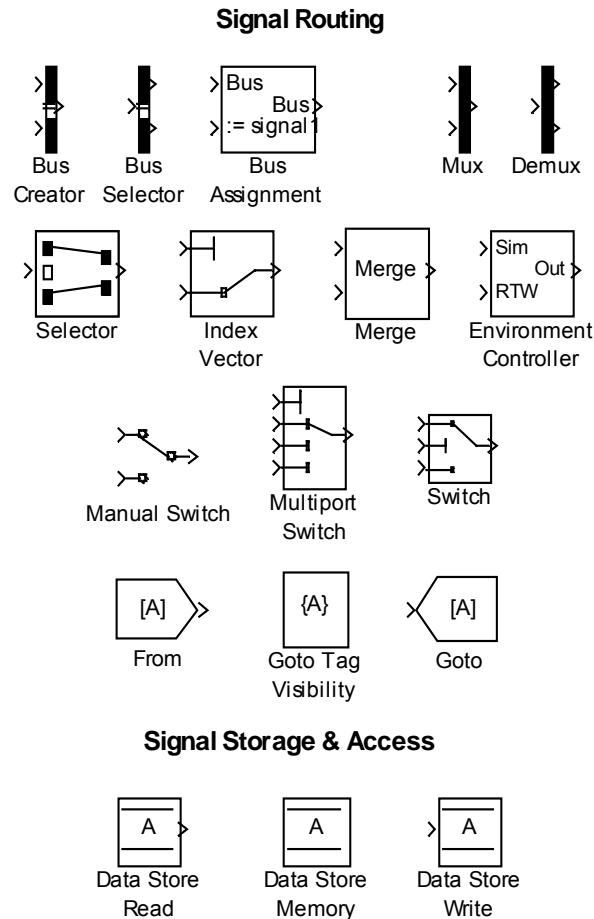



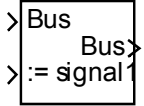



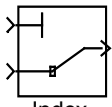


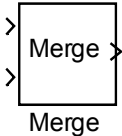
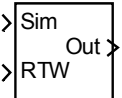
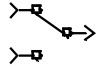
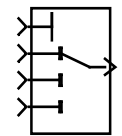
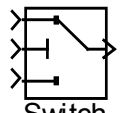
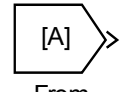
Рис. 3.7

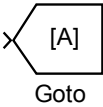

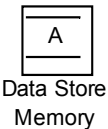
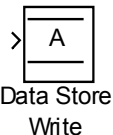

Таблица 3.7

Раздел Simulink| Signal Routing

Имя блока	Назначение, краткое описание	Пример модели
Bus Creator  Bus Creator	Блок Bus Creator – шинный формирователь – формирует шину из Number of inputs сигналов одинаковых или различных типов. Он позволяет объединять любые сигналы (векторные, матричные, комплексные, действительные и целочисленные разных типов) в единую шину. Сигналы в шине могут сохранять прежние имена (Inherit bus signal names from input ports) или переименовываться (Require input signal names to match signals below), перемещаться в шине (Up, Down). Список сигналов в шине отображается в поле Signals in bus.	
Bus Selector  Bus Selector	Блок Bus Selector – шинный селектор выделяет из шины требуемые сигналы. Список выделяемых Selected signals сиг-	

 <p>Bus Selector</p>	<p>налов формируется их выделением в окне Signals in the bus и нажатием кнопки Select >>. С помощью кнопок Up, Down, Remove выделенные в выходной шине сигналы могут перемещаться вверх, вниз или удаляться из нее. Флаг Output as bus позволяет объединить выходные сигналы в одну шину.</p>	
<p>Bus Assignment</p>  <p>Bus Assignment</p>	<p>Блок Bus Assignment предназначен для замены части сигналов в шине другими сигналами, которые подаются на порты (:=). Заменяемые сигналы выделяются в окне Signals in the bus, переносятся нажатием кнопки Select >> в окно Signals that are being assigned. При добавлении в это окно нового сигнала на графическом изображении блока появляется новый порт (:=) с соответствующим названием, куда должен подаваться замещающий сигнал. Этот сигнал должен быть того же типа, что и заменяемый.</p>	
<p>Mux</p>  <p>Mux</p>	<p>Блок Mux объединяет Number of Inputs скалярных и (или) векторных входных сигналов в один векторный сигнал. Размерность выходного вектора равна сумме размерностей входных векторов. Параметр Number of Inputs можно задавать как вектор с указанием числа элементов каждого входного вектора, например [2 1 3].</p>	
<p>Demux</p>  <p>Demux</p>	<p>Блок Demux разделяет входной векторный сигнал на Number of outputs отдельных составляющих. Number of outputs может задаваться числом или вектором, определяющими каким образом должен быть разгруппирован входной сигнал. Флаг Bus selection mode служит для объединения или разделения векторных сигналов. В этом режиме блок работает с векторами в целом, а не с отдельными их элементами.</p>	
<p>Selector</p>  <p>Selector</p>	<p>Блок Selector выбирает из вектора или матрицы (Input type) требуемые элементы. Список Source of element indices определяет источник индексов входного сигнала: Internal – выбираемые элементы задаются параметром Elements; External – задаются извне сигналом на входе E₁. Для матричного сигнала выбираемые элементы задаются номерами строк (Rows) и столбцов (Columns). Параметр Input port width определяет размерность входного вектора.</p>	
<p>IndexVector</p>  <p>Index Vector</p>	<p>Блок IndexVector подключает к выходу один из входов в соответствии с величиной постоянного сигнала, подаваемого на его управляющий вход. Блок является вариантом реализации рассматриваемого ниже блока Multiport Switch. Если флаг Use zero-based indexing включен номера входов отсчитываются с нуля, в противном случае – с единицы.</p>	

<p>Merge</p>  <p>Merge</p>	<p>Блок Merge объединяет Number of inputs входных сигналов в единый векторный сигнал. Флаг Allow unequal port widths разрешает использовать неодинаковую размерность входных сигналов. Параметр Input port offsets вектор, задающий смещение каждого входного вектора относительно начала выходного. Если этот параметр не задается, размерность выходного вектора будет равна длине входного максимального размера, и все входные вектора будут иметь нулевое смещение, т.е. их компоненты будут по очереди замещать друг друга.</p>	
<p>Environment Cntroller</p>  <p>Environment Controller</p>	<p>Этот блок передает на выход сигнал либо из входного порта Sim, либо из порта RTW (Real Time Workshop – подсистема MatLab, предназначенная для работы в режиме реального времени). Это позволяет использовать одну и ту же блок-диаграмму модели в двух вариантах: либо для проведения моделирования, либо для генерации кодов для работы физических устройств, к которым подключается модель.</p>	
<p>Manual Switch</p>  <p>Manual Switch</p>	<p>Блок Manual Switch является ручным переключателем, к выходу которого может подключаться один из двух входных сигналов. Переключение между входами осуществляется двойным щелчком мыши по изображению блока перед началом моделирования или в процессе его.</p>	
<p>Multiport Switch</p>  <p>Multiport Switch</p>	<p>Блок Multiport Switch – многопозиционный переключатель. Он подключает к выходу один из Number of input ports входных портов, номер которого задается величиной сигнала на управляющем входе. Флаг Use zero-based indexing определяет точку отсчета (0 или 1) номеров входных портов.</p> <p>Если параметр Number of input ports равен 1, то блок работает в варианте IndexVector.</p>	
<p>Switch</p>  <p>Switch</p>	<p>Блок Switch выполняет переключение входных сигналов по сигналу управления. Когда управляющий сигнал (он подается на средний вход) удовлетворяет условию Criteria for passing first input (Threshold – пороговый уровень), то к выходу подключается сигнал с первого входа, в противном случае - со второго.</p>	
<p>From</p>  <p>From</p>	<p>Блок приема сигнала From принимает сигнал от блока Goto. Совместное использование этих двух блоков обеспечивает передачу сигнала без линии связи. Сигнал, принимаемый блоком From от блока Goto, задается в поле Goto Tag, где указывается идентификатор сигнала Tag, установленный в соответствующем блоке Goto. После любого изменения пара-</p>	

	метров в блоке Goto, с которым связан блок From необходимо обновить информацию о сигналах, нажав кнопку Update Tags.	
<p>Goto</p> 	<p>Блок передачи сигнала Goto передает сигнал блоку From. Параметр Tag задает имя передаваемого сигнала. Оно должно совпадать с именем, установленным в поле Goto Tag блока From, которому передается данный сигнал. Совместное использование этих двух блоков обеспечивает передачу сигнала без линии связи. Параметр Tag Visibility (признак видимости сигнала) определяет зону передачи сигнала: local – в пределах локальной подсистемы; scoped - в пределах локальной подсистемы и подсистемах нижних уровней иерархии; global – в пределах всей модели.</p>	
<p>Goto Tag Visibility</p> 	<p>Блок Goto Tag Visibility определяет доступность сигналов, видимость которых в блоке Goto Tag определена опцией scoped, для блоков From подсистем нижних уровней иерархии. Для сигналов с зонами видимости local или global необходимость в применении этого блока отсутствует. Наименование сигнала, который делается видимым для блоков From подсистем нижних уровней иерархии, указывается в поле Goto tag</p>	
<p>Data Store Memory</p> 	<p>Блок Data Store Memory создает область памяти для хранения данных с именем, определяемым в поле Data store name. Блок используется совместно с блоками Data Store Write (запись данных) и Data Store Read (чтение данных). Параметр Initial value задает начальные значения размещаемых в памяти данных и может использоваться для задания их размерности. Например, если начальное значение сигнала задано матрицей [1 2; 3 4], то сохраняемый сигнал должен быть матрицей 2×2. Хранимые данные доступны в подсистемах нижних уровней иерархии.</p>	
<p>Data Store Write</p> 	<p>Блок Data Store Write записывает данные в область памяти Data store name, созданную блоком Data Store Memory с тактовым интервалом Sample time. Если Sample time = -1, то величина тактового интервала наследуется от предыдущего блока. В модели может быть несколько блоков Data Store Write, выполняющих запись в одну область памяти.</p>	
<p>Data Store Read</p> 	<p>Блок Data Store Read предназначен для считывания данных из области памяти Data store name, созданной блоком Data Store Memory, с тактовым интервалом Sample time. В модели может быть несколько блоков Data Store Read, выполняющих чтение данных из одной области памяти с одинаковым или разными тактовыми интервалами Sample time.</p>	

3.8. Sources - источники сигналов

Библиотека источников сигналов Sources содержит два подраздела: Model & Subsystem Inputs – модели и подсистемы-источники и Signal Generators – генераторы сигналов.

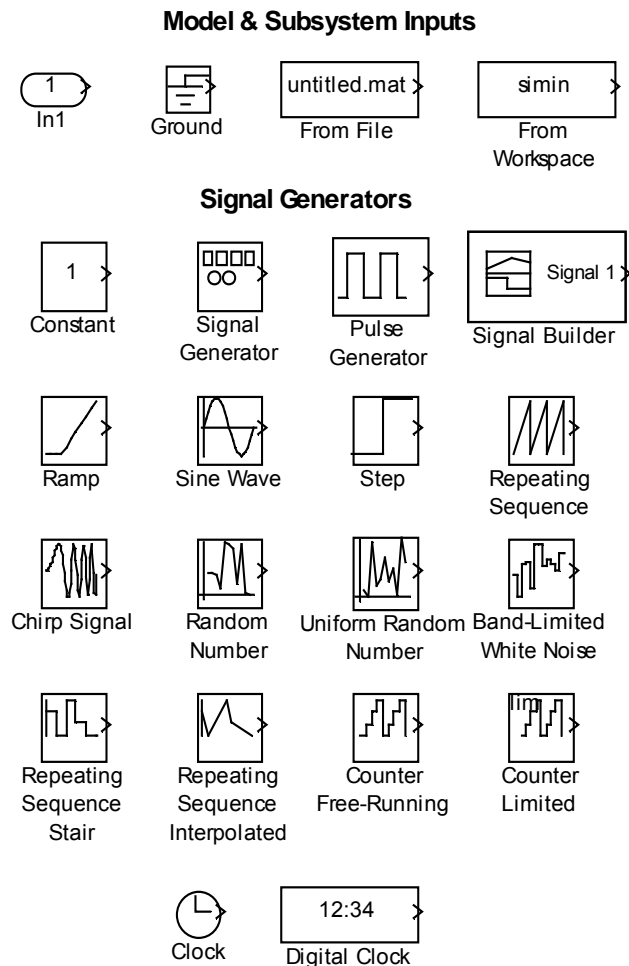


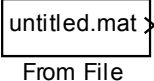


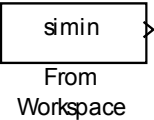
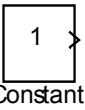
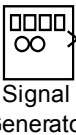
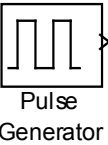
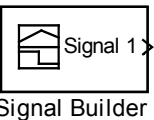
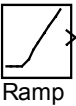
Рис. 3.8







Таблица 3.8







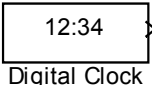
Раздел Simulink| Sources

Имя блока	Назначение, краткое описание	Пример модели
Inport	Блок Inport –входной порт – используется для создания входного порта подсистемы или модели верхнего уровня ие-	

	<p>рации. В первом случае блок используется для передачи данных из модели в подсистему, во втором – для передачи данных и сигналов из рабочей области (workspace) MatLab в модель. Параметры: Port number – номер порта; Port dimensions – размерность порта (если Port dimensions= -1, то размерность устанавливается автоматически по подключенному сигналу); Sample time - тактовый интервал, с которым сигнал считывается из порта; флаг Interpolate data позволяет установить режим интерполяции входного сигнала, когда временные отсчеты входного сигнала, считываемого из рабочего пространства, не совпадают с Sample time блока Inport.</p> <p>При создании подсистем и добавлении в них блоков Inport входные порты нумеруются автоматически, начиная с 1. При удалении блока Inport остальные порты перенумеровываются. Последовательность номеров портов должна быть непрерывной, иначе Simulink при моделировании выдаст сообщение об ошибке. В этом случае необходимо вручную перенумеровать их. В системе верхнего уровня иерархии блок Inport используется для передачи сигнала из рабочей области MatLab в модель. В этом случае помимо установки в модели входного порта необходимо в окне Configuration Parameters в разделе Data Import/Export активизировать флаг Input, указать имя передаваемого сигнала и выбрать его тип в поле Format.</p>	
<p>Ground</p> 	<p>Блок Ground (земля) можно использовать, чтобы "заземлить" входные порты блоков, не подключенные к другим блокам. Это блокирует вывод предупреждающих сообщений при моделировании. Блок Ground выдает нулевой сигнал на вход порта, к которому он подключен.</p>	Ground_.mdl
<p>From File</p> 	<p>Блок From File используется для чтения данных из внешнего mat-файла данных, указанного в поле File name. Такой файл может быть создан в Simulink с помощью блока To File. Если тактовый интервал данных в файле не совпадает с параметром, установленным в поле Sample time, блок выполняет их интерполяцию.</p>	<p>From_File.mdl</p> <p>Предварительно должен быть создан файл данных Example_file.mat (например, с помощью файла-примера To_File.mdl)</p>

	<p>Блок From Workspace используется для считывания данных из рабочей области (пространства) MatLab в модель. В поле Data должно быть указано имя переменной (матрицы или структуры), передаваемой из рабочего пространства в модель. Флаг Interpolate data позволяет установить режим интерполяции входного сигнала, когда временные отсчеты входного сигнала, считываемого из рабочего пространства, не совпадают с Sample time. Список Form output after final data value by позволяет указать форму выходного сигнала после того, как закончились считываемые данные.</p>	
<p>Constant</p> 	<p>Блок Constant генерирует действительную или комплексную постоянную величину, заданную в поле Constant value: скаляр, вектор (одномерный массив) или матрицу (двумерный массив) - в зависимости от формы ввода генерируемой константы в поле Constant Value.</p>	Const_.mdl
<p>Signal Generator</p> 	<p>Блок Signal Generator формирует один из четырех видов периодических сигналов Wave form: синусоидальный (sine), прямоугольный (square), пилообразный (sawtooth), случайный, случайный (random). Параметры: Amplitude – амплитуда, Frequency - частота, Units – единица измерения частоты (Hertz – Гц, rad/sec – рад/с)</p>	Sign_Gen.mdl
<p>Pulse Generator</p> 	<p>Блок Pulse Generator формирует периодическую последовательность прямоугольных импульсов с амплитудой Amplitude, периодом Period, относительной длительностью Pulse Width, задаваемой в % по отношению к периоду, и задержкой Phase Delay. Сигнал может формироваться двумя способами Pulse type: time-based (по текущему времени) или sample-based (по количеству тактовых интервалов Sample Time)</p>	Pulse_Gen.mdl
<p>Signal Builder</p> 	<p>Блок Signal Builder (конструктор сигналов) позволяет создавать в интерактивном режиме группу сигналов на основе кусочно-линейной аппроксимации их формы. Базовый тип сигнала выбирается с помощью главного меню (пункт Signal) и отображается в виде временной диаграммы, рисунок которой можно корректировать с помощью мыши. В блоке реализованы следующие базовые типы сигналов: Constant, Step, Pulse, Square..., Triangle..., Sampled Sin..., Sampled Gaussian Noise, Pseudorandom Noise, Poisson Random Noise.</p>	Sign_Build.mdl
<p>Ramp</p> 	<p>Блок Ramp формирует линейно изменяющийся во времени со скоростью Slope сигнал, который начинается в момент времени Start time с начального значения Initial output.</p>	Ramp_.mdl

Sine Wave  Sine Wave	<p>Блок Sine Wave формирует синусоидальный сигнал с амплитудой Amplitude и постоянной составляющей (смещением) Bias. Способ задания частоты и начальной фазы сигнала зависит от параметра Sine type - способа привязки формируемого сигнала ко времени: Time-based – к текущему времени и Sample-based – к величине шага модельного времени. В первом случае задаются круговая частота Frequency (рад/с) и начальная фаза Phase (рад). Во втором – количество отсчетов на период сигнала (Samples per period) и дискретный фазовый сдвиг (Number of offset samples - количество шагов модельного времени до нулевой фазы сигнала).</p>	Sine_Wave. mdl
Step  Step	<p>Блок Step (генератор единичного скачка) формирует скачок постоянного напряжения (тока) со значения Initial value до значения Final value в момент времени Step time.</p>	Step_.mdl
Repeating Sequence  Repeating Sequence	<p>Блок Repeating Sequence формирует периодически повторяющийся сигнал, задаваемый таблично. В поле Time values задается вектор временных отсчетов сигнала, а в поле Output values – вектор значений сигнала в отсчетные моменты времени. Оба вектора должны иметь одинаковую размерность. На интервалах времени между отсчетными значениями сигнал аппроксимируется линейной функцией.</p>	Repeat_Seq. mdl
Chirp Signal  Chirp Signal	<p>Блок Chirp Signal формирует гармонический сигнал с линейно-изменяющейся частотой от значения Initial frequency до значения Frequency at target time за время Target time.</p>	Chirp_Sig.mdl
Random Number  Random Number	<p>Блок Random Number формирует гауссов шум - случайный сигнал с нормальным распределением. Mean – среднее значение, Variance – дисперсия, Initial seed - любое целое неотрицательное число, с которого стартует Random Number. Для одинаковых чисел реализации случайного сигнала одинаковы.</p>	Rand_Numb. mdl
Uniform Random Number  Uniform Random Number	<p>Блок Uniform Random Number формирует случайный сигнал с равномерным распределением значений от Minimum до Maximum. Initial seed - любое целое неотрицательное число, с которого стартует Uniform Random Number. Для одинаковых чисел реализации случайного сигнала одинаковы.</p>	Rand_Numb. mdl
Band-Limited White Noise	<p>Блок Band-Limited White Noise формирует гауссов белый шум с мощностью Noise power и интервалом корреляции Sample time. Верхняя граничная частота шума определяется величиной Sample time: $F_{\max} = 0,01 / \text{Sample time}$, [Гц]. Если</p>	Rand_Numb. mdl

 Band-Limited White Noise	параметр Noise power вектор, то выходной сигнал тоже будет вектором. Чтобы компоненты вектора не были одинаковыми Initial seed также должно быть вектором.	
Repeating Sequence Stair  Repeating Sequence Stair	Блок Repeating Sequence Stair формирует т периодически повторяющийся ступенчатый сигнал, задаваемый вектором Vector of output values . Значения сигнала изменяются через интервал Sample time	Repeat_Seq. mdl
Repeating Sequence Interpolated  Repeating Sequence Interpolated	Блок Repeating Sequence Interpolated формирует периодически повторяющийся сигнал, задаваемый таблично. В поле Vector of time values задается вектор временных отсчетов сигнала, а в поле Vector of time values – вектор выходного сигнала в отсчетные моменты времени. Оба вектора должны иметь одинаковую размерность. На интервалах времени между отсчетными значениями сигнал интерполируется в соответствии с методом, задаваемым в поле Look-Up Method .	Rep_Int.mdl
Counter Free-Running  Counter Free-Running	Выходной сигнал блока Counter Free-Running - количество временных интервалов Sample time в диапазоне (0- $2^{\text{Number of Bits} - 1}$). При достижении состояния $2^{\text{Number of Bits} - 1}$ счетчик сбрасывается в нуль, и счет начинается сначала.	Counter.mdl
Counter Limited  Counter Limited	Выходной сигнал блока Counter Limited - количество временных интервалов Sample time в диапазоне (0 - Upper limit). При достижении состояния Upper limit счетчик сбрасывается в нуль, и счет начинается сначала.	Counter.mdl
Clock  Clock	Блок Clock формирует выходной сигнал, величина которого на каждом шаге расчета равна текущему времени моделирования. Параметр Decimation (положительное целое число) определяет интервал, через который изменяется сигнал на выходе блока.	Clock.mdl
Digital Clock  Digital Clock	Блок Digital Clock формирует выходной сигнал, величина которого на каждом шаге расчета равна текущему времени моделирования в тактовые интервалы времени Sample time . Внутри такта выходной сигнал блока не изменяется.	Clock.mdl

3.9. Sinks - приемники сигналов

Библиотека приемников сигналов Sinks содержит три подраздела: Model & Subsystem Outputs – модели и подсистемы-приемники сигналов, Data Viewers – приемники для наблюдения сигналов .

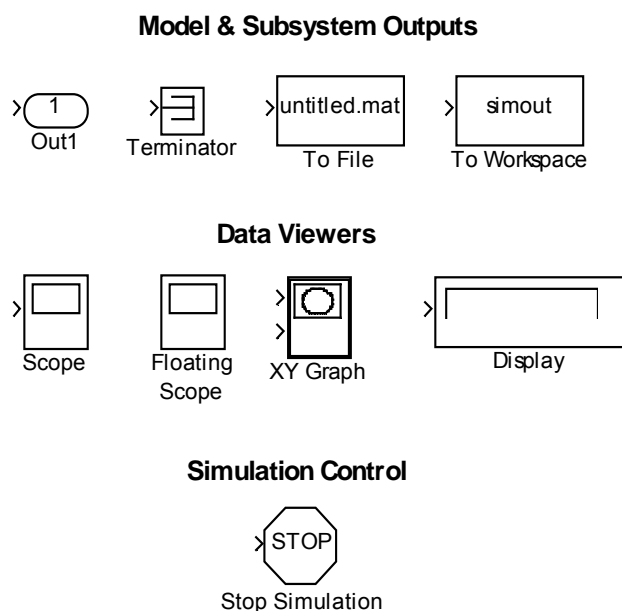


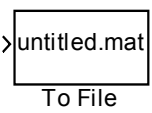
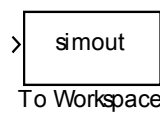


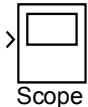


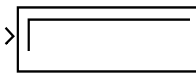
Рис. 3.9


Таблица 3.9

Раздел Simulink| Sinks

Имя блока	Назначение, краткое описание	Пример модели
Outport 	Блок Outport – выходной порт – используется для создания выходного порта подсистемы или модели верхнего уровня иерархии. В первом случае блок используется для передачи данных из подсистемы в модель, во втором – для передачи данных и сигналов из модели. В рабочую область (workspace) MatLab. Параметры: Port number – номер порта; Port dimensions – размерность порта (если Port dimensions = -1, то размерность устанавливается автоматически по подключенному сигналу); Sample time – тактовый интервал, с которым сигнал считывается из порта. Список Output when disabled становится доступ-	

	<p>ным, если блок Outport используется в управляемой подсистеме. Он определяет вид сигнала на выходе подсистемы, когда она не активна: held – выходной сигнал равен последнему рассчитанному значению; reset – значению, задаваемому параметром Initial output.</p> <p>При создании подсистем и добавлении в них блоков Outport выходные порты нумеруются автоматически, начиная с 1. При удалении блока Outport остальные порты перенумеровываются. Последовательность номеров портов должна быть непрерывной, иначе Simulink при моделировании выдаст сообщение об ошибке. В этом случае необходимо вручную перенумеровать их. В системе верхнего уровня иерархии блок Outport используется для передачи сигнала из модели в рабочую область MatLab. В этом случае помимо соответствующих установок в модели выходного порта необходимо в разделе Data Import/Export окна Configuration Parameters активизировать флаг Output, указать имя передаваемого сигнала и выбрать его тип в поле Format.</p>	
<p>Terminator</p>  <p>Terminator</p>	<p>Блок Terminator можно использовать как "заглушку" на выходной порт любого блока, не подключенный к другим блокам. Это позволяет подавить вывод предупреждающих сообщений о не использовании таких портов при моделировании.</p>	Term_.mdl
<p>To File</p> 	<p>Блок To File используется для записи данных в mat-файл, указанный в поле File name. По умолчанию файл имеет имя untitled.mat и, если не указан путь к файлу, размещается в текущей рабочей папке. Variable name – имя переменной, содержащей записываемые данные. Decimation шаг децимации записываемых данных. При Decimation=1 в файл записывается каждое значения входного сигнала, при Decimation=2 – каждое второе, при Decimation=3 – каждое третье и т.д. Каждый отсчет сигнала записывается в виде вектор-столбца, первый элемент которого содержит модельное время, а остальные вектор сигнала.</p>	To_File.mdl
<p>To Workspace</p> 	<p>Блок To Workspace используется для сохранения данных, поступивших на его вход, в рабочей области (workspace) под именем Variable name. Параметр Limit data points to last определяет предельное количество сохраняемых отсчетных точек сигнала от момента окончания моделирования, Decimation шаг децимации сохра-</p>	

	<p>няемых данных, например при Decimation=2 будет сохраняться каждая вторая точка. Список Save format определяет формат, в котором будут храниться данные.</p>	
 <p>Scope</p>	<p>Блок Scope (осциллограф, его аналоги: Floating Scope, Signal Viewer Scope) строит графики сигналов на его входе в функции от времени. Он предназначен для наблюдения за изменениями сигналов в процессе моделирования. Окно просмотра сигналов открывается двойным щелчком мыши по изображению блока в модели. Если на вход блока поступает векторный сигнал, график каждого компонента сигнала отображается разным цветом: 1-го – желтым, 2-го – сиреневым, 3-го – голубым, 4-го – красным, 5-го – зеленым, 6-го – темно голубым. Далее цвета повторяются. Настройка блока выполняется с помощью панели инструментов, которая содержит 12 кнопок. Подробнее см. п.1.8.</p>	Scope_.mdl
 <p>Floating Scope</p>	<p>Блок Floating scope - это блок Scope в “свободном” режиме. В этом режиме осциллограф не имеет входов, а выбор сигналов осуществляется с помощью инструмента Signal selection. С его помощью открывается окно Signal selector, и в нем отмечаются флажками имена блоков, выходные сигналы которых должны отображаться осциллографом. Блок удобно использовать для наблюдения сигналов в сложных моделях. Подробнее см. п.1.8.</p>	Scope_.mdl
<p>XY Graph</p>  <p>XY Graph</p>	<p>Блок XY Graph – графопостроитель, отображает графически зависимость $Y(X)$, где X и Y – сигналы, подаваемые на его входы. На верхний вход подается сигнал X, на нижний Y. Параметры: x-min, x-max и y-min, y-max – минимальное и максимальное значения сигналов по осям x и y соответственно.</p>	XY_Graph.mdl
<p>Display</p>  <p>Display</p>	<p>Блок Display отображает значение сигнала в виде числа. Формат отображения числа определяется параметром Format: short и long соответственно 5 и 15 значащих десятичных цифр; short_e и long_ соответственно 5 и 15 значащих десятичных цифр и 3 символа степени 10; bank – “денежный” формат. Формат с фиксированной точкой и двумя десятичными знаками в дробной части. Decimation шаг децимации отображаемых данных, например при Decimation=2 будет отображаться каждое второе значение сигнала. Флагом Floating display блок переводится в “свободный” режим: входной порт отсутствует, а отображаемый сигнал выбирается щелчком ле-</p>	Display_.mdl

	вой кнопки мыши по требуемой линии связи. Перед работой в этом режиме необходимо снять флаг Signal storage reuse в разделе Optimization окна Configuration Parameters, которое открывается из главного меню окна модели (пункт Simulation).	
<p>Stop</p>  <p>Stop Simulation</p>	Блок Stop останавливает расчет, когда сигнал на его входе становится отличным от нуля. Если входной сигнал векторный, для остановки моделирования достаточно, чтобы один из компонентов вектора стал не нулевым.	Stop_.mdl

3.10. Обзор библиотеки Simulink Extras

Библиотека Simulink Extras является дополнительной библиотекой пакета Simulink, но в отличие от ряда расширений этот пакет входит в состав его поставки. Эта библиотека содержит наборы блоков с более широкими функциями, чем рассмотренные выше разделы основной библиотеки. Однако, это не означает, что применение этой библиотеки всегда предпочтительнее. Связано это с тем, что усложнение функций блоков, полезное при решении ряда специфических задач, оборачивается усложнением моделирования при решении большинства обычных задач.

Область применения библиотеки Simulink Extras не очень широка. Блоки этой библиотеки будут рассмотрены во второй части пособия.

Библиографический список

1. Steven T. Karris. Introduction to Simulink with Engineering applications. Orchard Publications: [www. orchadpublications.com](http://www.orchadpublications.com), 2006.
2. Дьяконов В.П. MatLab 6.5 SP1/7 + Simulink 5/6. Основы применения. – М.: СОЛОН-Пресс, 2005.
3. Дьяконов В.П. Simulink 4. Специальный справочник. – СПб.: Питер, 2002.
4. Андриевский Б.Р., Фрадков А.Л. Избранные главы теории автоматического управления. – СПб: Наука, 2000
5. Черных И.В. Simulink среда создания инженерных приложений.- М.: Диалог-МИФИ, 2004.
6. Сергиенко А.Б. Цифровая обработка сигналов: Учебник для вузов. 2-е изд. – СПб.: Питер, 2006. –701 с.
7. Суворова Е. А., Шейнин Ю. Е. Проектирование цифровых систем на VHDL. – СПб: БХВ–Петербург, 2003.

Шеболков Виктор Васильевич

**МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ КОМПОНЕНТОВ РАДИО-
ЭЛЕКТРОННЫХ УСТРОЙСТВ И СИСТЕМ В ПАКЕТЕ SIMULINK**

Учебное пособие

Ответственный за выпуск Шеболков В.В.

Редактор Надточий З.И.

Корректор Надточий З.И.

ЛР №020565 от 23.06. 1997г. Подписано к печати _____ г.

Бумага офсетная. Печать офсетная.

Формат 60х84_{1/16}

Усл.п.л. – 3,7. Уч.-изд.л. 3,6.

Заказ №_____ Тираж экз.

_____”С”_____

Издательство Технологического института
Южного федерального университета
ГСП 17А, Таганрог, 28, Некрасовский, 44
Типография Технологического института
Южного федерального университета
ГСП 17А, Таганрог, 28, Энгельса, 1