Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Промышленная электроника»

В. В. Щуплов

МОДЕЛИРОВАНИЕ СИГНАЛОВ И СИСТЕМ В СРЕДАХ MATLAB И SIMULINK

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
по курсу «Теория электросвязи»
для студентов специальности 1-36 04 02
«Промышленная электроника»
дневной формы обучения
В двух частях
Часть 1

Гомель 2010

УДК 621.391:004.94(075.8)
ББК 32.811.3В6я73
Щ97

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем ГГТУ им. П. О. Сухого
(*протокол № 4 от 08.12.2009 г.*)

Рецензент: канд. техн. наук, доц. каф. «Электроснабжение»
ГГТУ им. П. О. Сухого *О. Г. Широков*

Щуплов, В. В.

Щ97 Моделирование сигналов и систем в средах Matlab и Simulink : лаборатор. практикум по курсу «Теория электросвязи» для студентов специальности 1-36 04 02 «Промышленная электроника» днев. формы обучения : в 2 ч. Ч. 1 / В. В. Щуплов. – Гомель : ГГТУ им. П. О. Сухого, 2010. – 30 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://lib.gstu.local. – Загл. с титул. экрана.

Даны необходимые сведения для освоения теоретического материала и практического закрепления знаний по курсу «Теория электросвязи». Рассмотрены практические примеры моделирования сигналов, дискретизированных по времени, и вопросы моделирования динамической системы при воздействии на нее информационного сигнала и помех.

Для студентов специальности 1-36 04 02 «Промышленная электроника» дневной формы обучения.

УДК 621.391:004.94
ББК 32.811.3В6я73

Лабораторная работа № 1
Моделирование динамических систем с помощью модуля Simulink из пакета MATLAB

Цель работы: ознакомиться с моделированием динамических процессов и инженерных систем связи с помощью модуля Simulink из пакета MATLAB на примере простого дифференциального уравнения, описывающего динамическую систему при воздействии сигналов и помех.

Блоки Программы Simulink (справка)

Блок **Algebraic Constraint** (Алгебраическое ограничение) (библиотека Math Operations
(Математические действия)). Инструмент для решения уравнений, похож на команду **fzero**.

Блок **Clock** (Часы) (библиотека Sources (Источники)). Выдает время при моделировании.

Блок **Constant** (Постоянная) (библиотека Sources (Источники)). Выдает определенную постоянную величину (константу).

Блок **Demux** (Разделитель) (Библиотека Signal Routing (Разводка сигнала)). Разбирает векторный сигнал на составляющие.

Блок **From Workspace** (Из рабочего пространства) (библиотека Sources (Источники)). Получает входной параметр из рабочего пространства среды MATLAB.

Блок **Gain** (Увеличение) (библиотека Math Operations (Математические действия)). Умножает на постоянную величину или на постоянную матрицу.

Блок **Integrator** (Итегратор) (библиотека Continuous (Длительные)). Вычисляет определенный интеграл, используя указанное начальное условие.

Блок **Math** Function (Математическая функция) (библиотека Math Operations (Математические действия)). Вычисляет показательные функции, логарифмы и т. п.

Блок **Mux** (Объединитель) (Библиотека Signal Routing (Разводка сигнала)). Собирает скалярные сигналы в векторный сигнал.

Блок **Polynomial** (Полином) (библиотека Math Operations

(Математические действия)). Вычисляет полиномиальную функцию.

Блок **Product** (Произведение) (библиотека Math Operations (Математические действия)). Умножает или делит сигналы. Может также инвертировать матричные сигналы.

Блок **Ramp** (Пилообразный сигнал) (библиотека Sources (Источники)). Создает функцию, которая сначала постоянна, а затем, начиная с определенного времени, линейно увеличивается.

Блок **Scope** (Индикатор) (библиотека Sinks (Приемники)). Строит график сигнала, как функцию от времени.

Блок **Sine Wave** (Синусоида) (библиотека Sources (Источники)). Создает синусоидальные колебания. Вы можете отрегулировать амплитуду, частоту и фазу колебаний.

Блок **Sum** (Суммирование) (библиотека Math Operations (Математические действия)). Складывает или вычитает входные данные.

Блок **To Workspace** (В рабочее пространство) (библиотека Sinks (Приемники)). Выводит сигнал в рабочее пространство среды MATLAB.

Блок **Trigonometric Function** (Тригонометрическая функция) (библиотека Math Operations (Математические действия)). Вычисляет тригонометрические или гиперболические функции.

Блок **Unit Delay** (Единичная задержка) (библиотека Discrete (Дискретные элементы)). Этот блок может оказаться полезным в моделировании с приращениями или в дифференциальных/разностных уравнениях.

Блок **XY Graph** (График XY) (библиотека Sinks (Приемники)). Строит график одного сигнала в зависимости от другого.

Ход работы

Запустить программу MATLAB. После запуска программы MATLAB запустить программу **Simulink** дважды щелкнув мышью на ярлыке Simulink панели запуска, либо щелкнув мышью на кнопке **Simulink,** расположенной на панели инструментов **Рабочего стола** программы MATLAB, или просто введя команду **simulink** в окне **Command Window** (Командное окно). При этом открывается окно **Simulink library** (Библиотека Simulink). В системах семейства

Windows открывается окно **Simulink Library Browser** (Обзор библиотеки Simulink), показанное на рис.1.1.

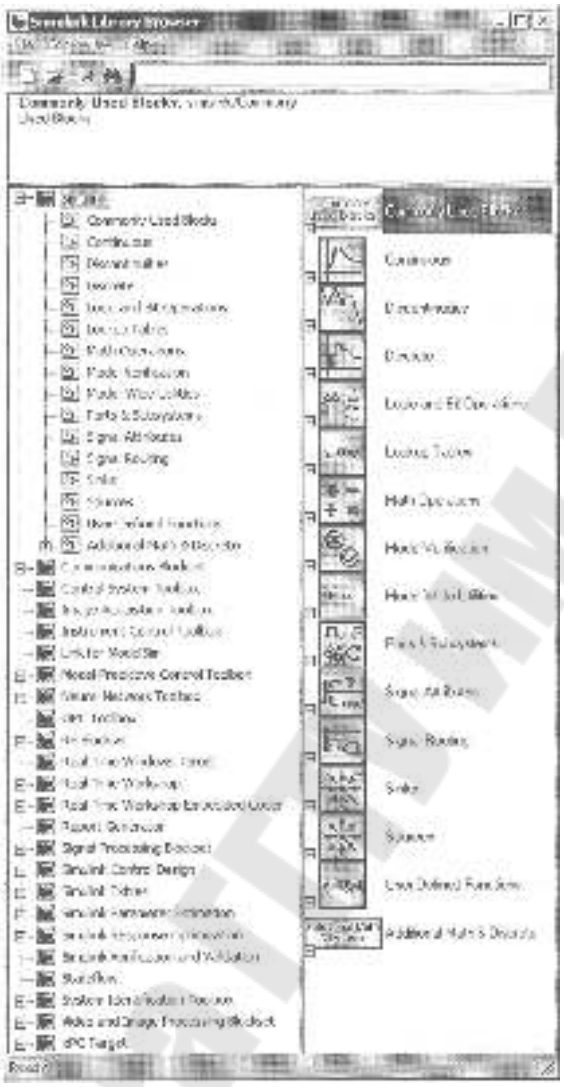


Рис. 1.1. Окно **Simulink Library Browser**

1. Модель с однородным дифференциальным уравнением.

1)Чтобы начать использование программы Simulink, выберите команду **File ♦ New ♦ Model** (Файл ♦ Создать ♦ Модель). При этом откроется пустое окно модели. Модель в программе Simulink создается путем копирования элементов, называемых блоками, из различных библиотек в окно модели. Мы объясним, как используется эта процедура, на примере модели однородного простого линейного дифференциального уравнения (ODE) $\boldsymbol{u'' + 2u' + 5u = 0}$, которое отображает работу излучателя затухающего гармонического сигнала.

Сначала нам необходимо уяснить, как представить уравнение в виде, в котором программа Simulink сможет его моделировать. Вот один из таких способов. Так как временная переменная является постоянной (непрерывной), сначала мы открываем библиотеку **Continious** (Постоянные) в системе **Windows** - либо щелкнув мышью на маленьком значке + слева от значка **Continious(Постоянные)** в правой верхней части окна (рис. 1.1*),* или щелкнув мышью на маленьком значке слева от слова «Continious» (Постоянные) в левой части окна **Simulink Library Browser** (Обзор библиотеки Simulink). В системе Windows после открытия библиотека **Continious** (Постоянные) выглядит так (рис. 1.2).

Рис. 1.2. Библиотека **Continious** (Постоянные)

Обратите внимание, что ***u*** и ***u′*** получаются из ***u′*** и ***u″*** (соответственно) путем интегрирования. Поэтому перетащите с помощью мыши две копии блока **Integrator** (Интегратор) в окно модели. Программа **Simulink** автоматически присвоит второму блоку имя **Integrator1** (Интегратор 1), чтобы отличить его от первого блока с именем **Integrator** (Интегратор).

Обратите внимание, что каждый блок **Integrator** (Интегратор) имеет входной и выходной порты. Подровняйте выходной порт блока **Integrator** (Интегратор) с входным портом блока **Integrator1** (Интегратор 1) и соедините их стрелкой, используя левую кнопку мыши. Стрелка, соединяющая два блока, называется сигналом.

Дважды щелкните мышью на этой стрелке, и появится небольшое текстовое поле, в которое вы можете ввести название ***u′***. (Вы можете также щелкнуть правой кнопкой мыши на стрелке, выбрать команду **Signal Properties**... (Свойства сигнала) и ввести свое название в поле **Signal Name** (Название сигнала).)

2)Способом, описанным выше, добавьте стрелку, входящую в блок Integrator (Интегратор) (представляет ***u″***), и стрелку, выходящую из блока Integrator 1 (Интегратор 1) (представляет ***u***). Эти стрелки еще не обеспечивают соединение с другими блоками, поэтому программа Simulink отмечает их пунктирными линиями, чтобы напомнить вам, что они не являются полностью работоспособными. Идея состоит в том, что u получается путем интегрирования из ***u′***, а ***u′*** получается тем же путем из ***u″***. Ваше окно модели теперь должно выглядеть так, как изображено на рисунке ниже (рис. 1.3).

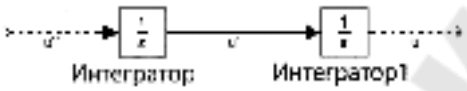


Рис. 1.3. Первая стадия модели в программе **Simulink**

3)Теперь нам необходимо использовать дифференциальное уравнение, которое можно записать в форме ***u″ = -5u – 2u′***. Нам необходимо добавить другие блоки, чтобы связать ***u″***, ввод первого блока **Integrator** (Интегратор), с ***u*** и ***u′*** соответственно этому уравнению. Для этой цели мы добавляем два блока **Gain** (Увеличение), которые выполняют умножение по константе, и один блок **Sum** (Сумма), используемый для сложения. Все они выбираются из библиотеки **Math Operations** (Математические операции) (седьмая сверху на рис. 1.2). После их извлечения (тем же способом, что и блоки **Integrator** (Интегратор)) мы получим окно модели, которое выглядит так (рис. 1.4).

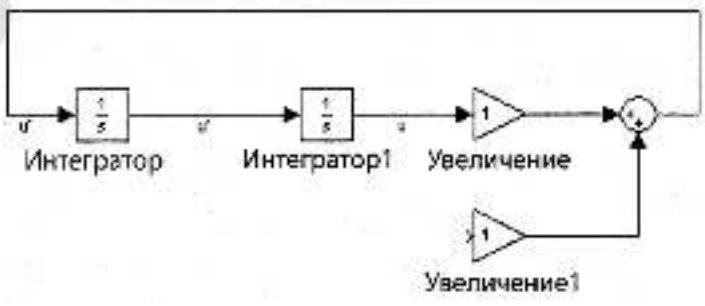


Рис. 1.4. Вторая стадия модели в программе Simulink

4)Теперь необходимо вернуться назад и отредактировать свойства блоков **Gain** (Увеличение), чтобы изменить константы, на которые они умножаются, со значения по умолчанию 1 на -5 (блок **Gain** (Увеличение)) и -2 (блок **Gainl** (Увеличение 1)). Для этого по очереди щелкните мышью на каждом из блоков **Gain** (Увеличение). Откроется диалог **Block Parameters** (Параметры блока), в котором можно изменять параметры, как вам требуется. Далее, нам необходимо направить ***u'***, вывод первого блока **Integrator** (Интегратор), к порту ввода блока **Gainl** (Увеличение 1). При этом возникает проблема, так как блок **Integrator** (Интегратор) имеет только один порт вывода, и он уже присоединен к следующему блоку. Поэтому нам необходимо познакомиться с разветвлением линий.

5)Поместите указатель мыши на середине стрелки, соединяющей два блока **Integrator** (Интегратор), нажмите и не отпускайте клавишу **Ctrl** одной рукой, одновременно нажмите левую кнопку мыши другой рукой, и перетащите указатель мыши к порту ввода блока **Gainl** (Увеличение 1). Мы почти закончили, но нам нужен еще блок для просмотра вывода. Откройте библиотеку **Sinks** (Приемники) и перетащите копию блока **Scope** (Экран) в окно модели. С помощью разветвления линии соедините этот блок с линией, соединяющей блоки **Integrator1** (Интегратор 1) и **Gain** (Увеличение). Здесь вам может понадобится переделать названия блоков (путем редактирования текста под каждым блоком), а также названия некоторых стрелок. В результате получится модель, показанная на рисунке ниже (рис. 1.5).

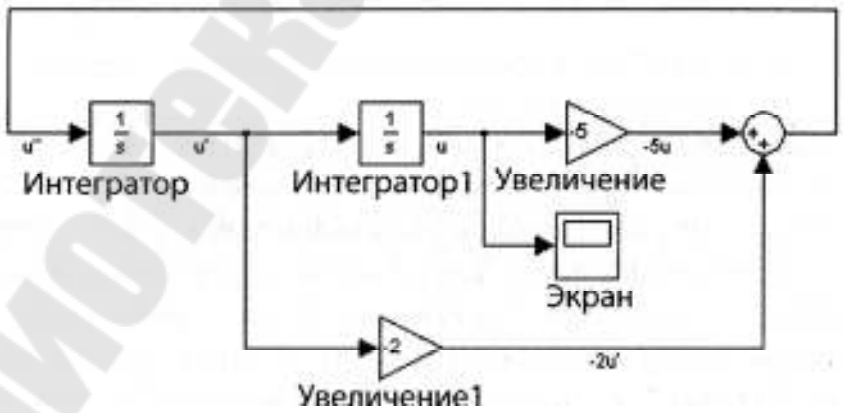


Рис. 1.5. Модель для уравнения ***u '' = -5u - 2 u'***
в программе **Simulink**

Теперь все готово для запуска процесса симуляции. Сначала следует сохранить модель, используя команду меню **File ♦ Save as...** (Файл ♦ Сохранить как). Модели можно присвоить имя **dampedosc.** (Программа MATLAB автоматически добавляет расширение файла **.mdl,** предназначенное для моделей.) Чтобы видеть, что происходит во время симуляции, дважды щелкните мышью на блоке **Scope** (Экран), чтобы открыть экран «осциллографа», на котором ***u*** будет изображаться как функция от ***t***. Разумеется, необходимо также задать начальные условия; это можно сделать, дважды щелкнув мышью на блоках **Integrator** (Интегратор) и изменив строку **Initial Condition** (Начальное условие) в диалоге **Block Parameters** (Параметры блока). Например, предположим, что мы задаем начальное условие для ***u'*** (в первом блоке **Integrator** (Интегратор)), как значение 5, и условие для ***u*** (во втором блоке **Integrator1** (Интегратор 1)), как значение 1. Другими словами, мы решаем систему линейных уравнений

$$\begin{aligned} u'' + 2u' + 5u &= 0; \\ u(0) &= 1; \\ u'(0) &= 5; \end{aligned}$$

которая имеет точное решение

$$u(t) = 3 \cdot e^{-t} \sin(2 \cdot t) + e^{-t} \cos(2 \cdot t) \ .$$

6)Перейдите в меню **Simulation** (Симуляция) и выберите команду **Start** (Запуск). В окне **Scope** (Экран) вы должны увидеть нечто похожее на то, что изображено на Рис. 1.6. Разумеется, это просто график функции $u(t) = 3 \cdot e^{-t} \sin(2 \cdot t) + e^{-t} \cos(2 \cdot t)$. (Кстати, вам может понадобится изменить масштаб на вертикальной оси в окне **Scope** (Экран). Щелчок мышью на значке с изображением бинокля производит автоматическое перемасштабирование, а щелчок правой кнопкой мыши на вертикальной оси открывает меню **Axes Properties...** (Свойства осей), которое позволяет вам выбирать вручную минимальные и максимальные значения зависимой переменной.) Если необходимо, то можно легко вернуться обратно, изменить некоторые параметры и снова перезапустить симуляцию.

Замечание. Возможно, вы сочтете, что лучше полагаться на блок Derivative (Производная), чем на блок Integrator (Интегратор), для симуляции дифференциальных уравнений.

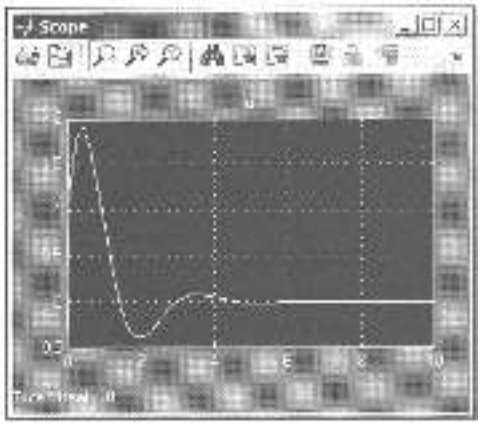


Рис. 1.6. Решение уравнения $u'' = -5u - 2u'$, $u(0) = 1$, $u'(0) = 5$ в окне **Scope** (Экран)

Но у этого пути есть две отрицательные стороны: сложнее установить начальные условия, а также числовое дифференцирование менее надежно, чем числовое интегрирование.

2.Модель с однородным дифференциальным уравнением.

Наконец, предположим, что теперь требуется изучить неоднородное уравнение для принудительных колебаний

$$u''+2u' + 5u = g(t),$$

где g является принудительно заданным периодом. Все, что нам надо для этого сделать - это добавить другой блок к модели из библиотеки **Sourses** (Источники). Щелкните мышью на черенке стрелки вверху модели, перемещаясь к первому блоку **Integrator** (Интегратор), и выберите команду меню **Edit ♦ Cut** (Редактирование ♦ Вырезать), чтобы удалить ее. Затем перетащите другой блок **Sum** (Сумма) (из библиотеки **Math Operations** (Математические операции)), поместите его перед первым блоком **Integrator** (Интегратор) и подведите подходящий источник к порту ввода блока **Sum** (Сумма). Например, если $g(t)$ представляет «шум», перетащите блок **Band-Limited White Noise** (Белый шум с ограниченной полосой частот) из библиотеки **Sourses** (Источники) на модель и подключите все так, как показано на рис. 1.7.

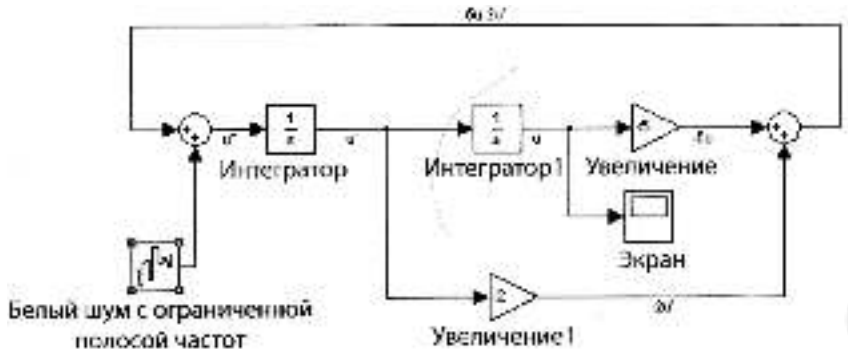


Рис. 1.7. Модель уравнения $u'' + 2u' + 5u = g(t)$ в программе **Simulink**

Выходной сигнал этой исправленной модели (со значениями по умолчанию 0.1 для поля **Noise power** (Мощность шума) и 0.1 для поля **Sample time** (Эталонное время)) показан на рисунке (рис. 1.8). Эффект шума в системе ясно различим в процессе симуляции.

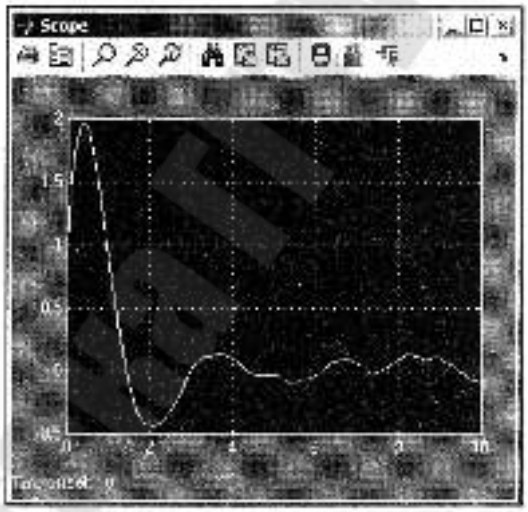


Рис. 1.8. Решение простого дифференциального уравнения для шума в окне **Scope** (Экран)

Задание для самостоятельной работы

1. Смоделировать систему, описываемую однородным дифференциальным уравнением, порядок и вид которого заданы преподавателем.
2. Смоделировать систему, описываемую неоднородным дифференциальным уравнением, порядок и вид которого заданы преподавателем.

Содержание отчета:

Отчет должен содержать цель работы, краткие теоретические сведения по принципу моделирования динамических систем в программе **Simulink**, описываемых линейными дифференциальными уравнениями, листинги выполнения работы с приведением моделей и графиков сигналов в характерных точках модели.

Контрольные вопросы:

1. Назовите основные блоки программы **Simulink**.
2. Как производится запуск программы **Simulink**.
3. Как получить производные от переменной.
4. Как сохранить модель.
5. Как задать сигнал и шум.
6. Как соединить блоки модели.
7. Как изменить масштаб на вертикальной оси в окне **Scope** (Экран).
8. Как запустить процесс симуляции.
9. Почему лучше использовать блок **Integrator** (Интегратор) для симуляции дифференциальных уравнений.
10. Как изменить константы блоков **Gain** (Увеличение).
11. Как выбирать вручную минимальные и максимальные значения зависимой переменной.

Лабораторная работа № 2

Моделирование дискретных сигналов в Matlab и Simulink

Цель работы: освоение приемов моделирования дискретных сигналов в средах Matlab и Simulink

Теоретические сведения. Дискретные сигналы в Matlab и Simulink

Вводные замечания

Подавая электрический сигнал с выхода микрофона на вход звуковой платы компьютера, полезно представлять себе, как аналоговый сигнал преобразуется в дискретный, и как затем дискретный сигнал преобразуется в последовательность чисел. В данном разделе мы рассмотрим первый этап – преобразование аналогового сигнала в дискретный. Такое преобразование принято называть «дискретизацией».

Возможные варианты сигналов показаны на рис. 2.1. Сигнал, изображенный на рис. 2.1.а, будем называть исходным аналоговым. На рис. 2.1.б представлена дискретная версия исходного сигнала, обычно именуемая данными, оцифрованными естественным способом, или данными с амплитудно-импульсной модуляцией (pulse amplitude modulation — PAM). Данные на рис.1.б еще несовместимы с цифровой системой, поскольку амплитуда каждой естественной выборки все еще может принимать бесконечное множество возможных значений, а цифровая система работает с конечным набором значений. На рис. 2.1.в и рис. 2.1.г показано представление исходного сигнала такими дискретными импульсами, вершина которых плоская. Если значения импульсов образуют несчетное множество, они называются дискретными отсчетами. Далее эти импульсы можно подать на устройство квантования, преобразующее импульсы так, что их значения образуют счетное множество - такие импульсы называются квантованными отсчетами. Данные в таком формате совместимы с цифровой системой.

Импульсы рис. 2.1.г отличаются от импульсов рис. 2.1.в тем, что полностью заполняют промежуток между моментами обновления значения сигнала. Такой способ дискретизации, именуемый «выборка-хранение», наиболее эффективен с точки зрения помехоустойчивости.

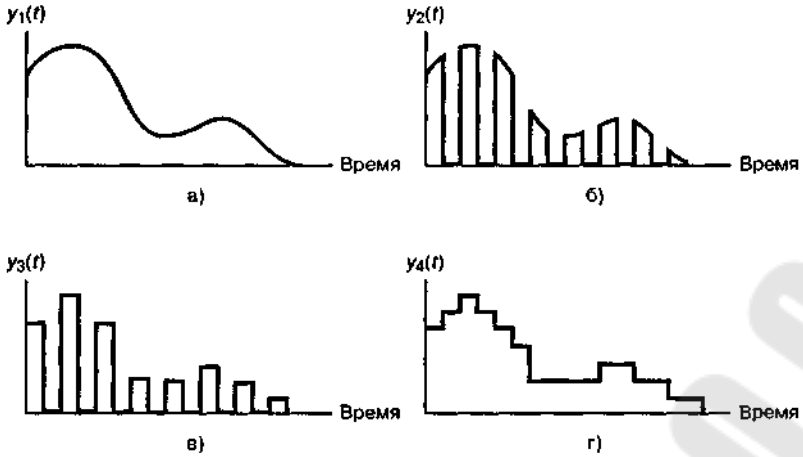


Рис. 2.1. Исходные данные в системе координат "время-амплитуда": а) исходный аналоговый сигнал; 6) данные в естественной дискретизации; в) квантованные выборки; г) выборка-хранение

1. Моделирование дискретных сигналов в Matlab

Генерировать сигналы в **Matlab** можно тремя способами:

1) в диалоговом режиме, с помощью последовательности команд в командном окне;
2) в автоматическом режиме, путем создания и запуска на выполнение m-скрипта;
3) в автоматическом режиме, путем создания и вызова m-функции.

Генерирование сигналов в диалоговом режиме. Этот способ наиболее трудоемок, поскольку требует каждую команду набирать с клавиатуры в командном окне. Чтобы повысить производительность труда, можно всю последовательность команд предварительно набрать в любом текстовом редакторе (обычно это **Notebook** или **Word**), а затем, скопировав текст в буферную память (**Clipboard**), вставить его в командное окно. Недостаток этого способа в том, что необходимо одновременно держать активными две программы – **Matlab** и текстовый редактор. Достоинство данного способа проявляется тогда, когда работу в **Matlab** производят, следуя некоей инструкции, в которой теоретические сведения чередуются с практическими заданиями в виде фрагментов текстов m-скриптов. Такой стиль работы типичен, например, при проведении лабораторных работ.

Например, так выглядит в текстовом редакторе последовательность команд генерирования **N** отсчетов тонального сигнала амплитудой **A**, частотой **f0**, начальной фазой **Fi0**, с частотой дискретизации **fs**:

```
% гармонический сигнал
A=1; f0=100; Fi0=pi/2; fs=1000; N=20;  % параметры сигнала
t=(0:N-1)/fs;                          % моменты времени
s=A*sin(2*pi*f0*t+Fi0);        % вычисление отсчетов сигнала
plot(t,s)                              % вывод графика
title('Гармонический сигнал')     % заголовок
xlabel('ВремЯ, с'); ylabel('Уровень');   % надписи вдоль осей
grid on                                % координатнаЯ сетка
```

Полученный график отображается в специальном окне с надписью Figure #1 (если это первый строящийся график). График удобно сохранять путем экспорта в экономном формате *.jpg (рис. 2.2).

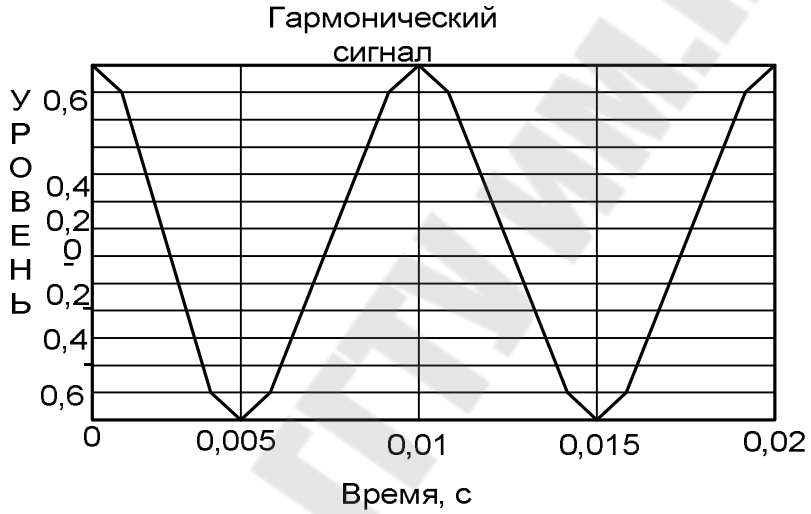


Рис. 2.2

Примечание: при использовании символов кириллицы в тексте команд (комментарии, заголовки и т.п.) следует учитывать особенности «отношения» каждой конкретной версии Matlab к кириллице. Так, в Matlab версии 6.1 нельзя употреблять строчную букву «я» - вместо нее следует писать прописную букву «Я». Именно по этой причине в тексте на рис. 2.2 вместо «Время» получилось «ВремЯ». Впрочем, эту надпись можно отредактировать (кнопка со стрелкой Edit Plot в графическом окне) перед тем, как сохранять рисунок на диске.

Генерирование сигналов путем создания m-скрипта. Данный способ отличается тем, что все команды набираются в специальном окне редактора m-файлов (рис. 2.3).

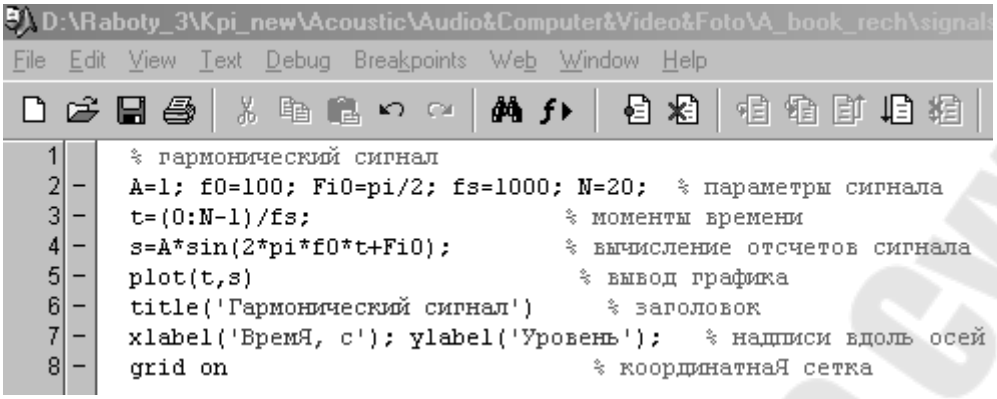


Рис. 2.3

Данный способ хорош тем, что вместо сторонних программных продуктов используется собственный инструментарий Matlab, специализированный для написания и отладки m-скриптов.

Генерирование сигналов путем создания m-функции. Данный способ отличается тем, что входные данные записывают как аргумент некоей функции $y = f(x)$, а выходные – как значение этой функции. Удобство в том, что символьные обозначения данных могут отличаться от обозначений, используемых в теле функции. Более того, числовые значения входных данных можно просто задавать в наименовании вызываемой функции. Последнее обстоятельство продемонстрируем на примере.

Создадим подпрограмму - m-скрипт **ton.m** вида:

```
% скрипт ton
s=A*sin(2*pi*f0*t+Fi0);        % вычисление отсчетов сигнала
```

Команду выполнения этого скрипта нужно «окружить» командами подготовки входных данных и вывода выходных данных:

```
A=1; f0=100; Fi0=pi/2; fs=1000; N=20;  % параметры сигнала
t=(0:N-1)/fs;                  % моменты времени
ton;                           % вычисление отсчетов сигнала
plot(t,s)                      % вывод графика
title('Гармонический сигнал')  % заголовок
xlabel('ВремЯ, с'); ylabel('Уровень');  % надписи вдоль осей
grid on                        % координатнаЯ сетка
```

Очевидно, обозначения входных и выходных данных вызывающей программы должны совпадать с обозначениями соответствующих данных вызываемой подпрограммы.

Теперь поступим по-иному – напишем и сохраним m-функцию

под именем **ton_sig.m**:

```
%-------функциЯ ton_sig.m -----------
% [s,t]=ton_sig(B,f1,Fi1,Fs,N1)
%-------------------------------------
% генерирование гармонического сигнала
% y = B * sin(2*pi*f1*x + Fi1),
% B - амплитуда;
% N1 - количество отсчетов сигнала;
% f1 - частота;
% Fs - частота дискретизации;
% Fi1 - начальнаЯ фаза сигнала
%-------------------------------------
function [y,x] = ton_sig( B, f1, Fi1, Fs, N1 )
%-------------------------------------
x = (0:N1-1)/Fs;    % моменты времени
y = B * sin( 2*pi*f1*x + Fi1 );
%-------- конец функции ton_sig.m ---------
```

Теперь m-скрипт генерирования того же отрезка косинусоиды будет выглядеть так:

```
% гармонический сигнал
[s,t]=ton_sig(1,100,pi/2,1000,20)  % вычисление отсчетов сигнала
plot(t,s)                  % вывод графика
title('Гармонический сигнал')     % заголовок
xlabel('ВремЯ, с'); ylabel('Уровень');   % надписи вдоль осей
grid on                    % координатнаЯ сетка
```

Как видим, теперь числовые значения входных данных задаются как аргументы m-функции **ton_sig.m**. Выходные данные функции используются для построения графика.

Очевидно, применение m-функций выгодно тогда, когда алгоритм формирования значений функции достаточно сложный: содержится много команд и обращений к разнообразным библиотечным функциям с непростым синтаксисом.

Очевиден и недостаток m-функций – необходимо помнить их синтаксис. Впрочем, получить нужную информацию можно, если в командном окне задать команду **help**:

```
>> help ton_sig
```

В результате на мониторе отобразится комментарий, с которого начинается m-функция. Для приведенного выше примера текст помощи имеет следующий вид:

```
%-------функциЯ ton_sig.m ----------
% [s,t]=ton_sig(B,f1,Fi1,Fs,N1)
%-------------------------------------
% генерирование гармонического сигнала
% y = B * sin(2*pi*f1*x + Fi1),
% B - амплитуда;
% N1 - количество отсчетов сигнала;
% f1 - частота;
% Fs - частота дискретизации;
% Fi1 - начальнаЯ фаза сигнала
%-------------------------------------
```

Таким образом, очевиден вывод: очень важно при программировании m-функций снабжать их качественным и подробным комментарием.

2. Моделирование дискретных сигналов в Simulink

Генерирование сигналов в **Simulink**, естественно, имеет свои особенности. Рассмотрим их.

Возьмем из библиотеки блоков **Simulink** два блока: **Sine Wave** (из раздела **Sources**) и **Scope** (из раздела **Sinks**). Соединив их, получим немудреную схему (рис. 2.4).

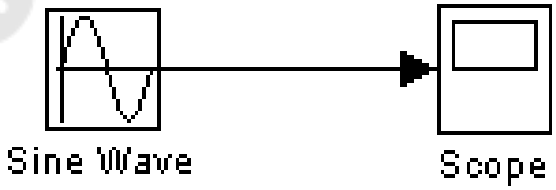


Рис. 2.4

Затем двойным щелчком по блоку осциллоскопа активизируем окно, имитирующее экран осциллоскопа, и запустим модель (кнопка **Start simulation**). В результате получим изображение отрезка синусоиды (рис. 2.5).

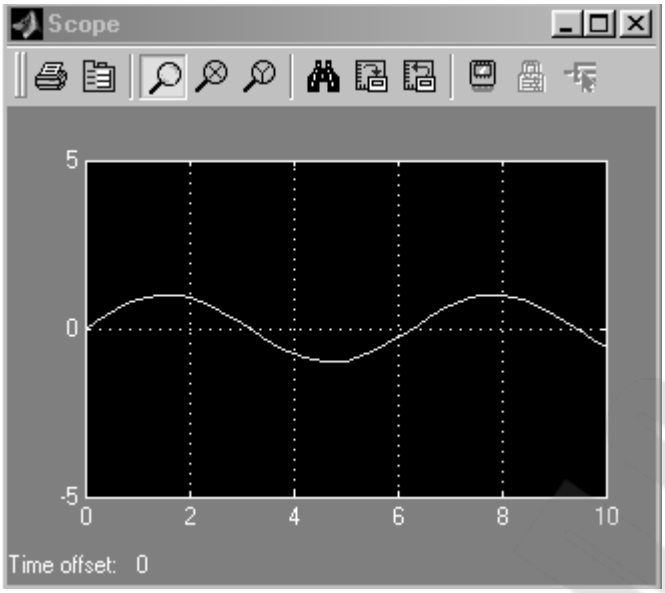


Рис. 2.5

Как видим, генерировать гармонический сигнал в среде Simulink даже проще, чем в среде Matlab. Однако это первое впечатление весьма обманчиво. Действительно, ведь важно еще уметь управлять параметрами гармонического сигнала. То, что амплитуда гармонического сигнала оказалась равной единице – нам просто «повезло». Действительно, по умолчанию амплитуда генерируемого сигнала принята равной единице. Однако частотой, начальной фазой и длительностью сигнала мы пока не управляем.

Дважды щелкнем по блоку **Sine Wave** – в результате появится окно настроек параметров (рис. 2.6). Щелкнув по кнопке **Help**, получим инструкцию по данному блоку, сущность которой сводится вкратце к тому, что в данном блоке выполняется операция

$$y = Amplitude \times \sin(frequency \times time + phase) + bias$$

Из приведенной формулы и надписей на рис. 2.6 становится понятным смысл четырех переменных: амплитуды, угловой частоты, начальной фазы и постоянной составляющей. Остается пока зашифрованным смысл переменной “время”.

Останавливаясь на этом важном вопросе, отметим различие понятий “время” и “модельное время”. Так, генерирование отрезка сигнала длительностью 1 с (модельное время) может длиться значительно более короткий промежуток времени, например, 0,1 с (реальное время). Скорость генерирования зависит от объема вычислений, быстродействия компьютера, от выбранного “решателя”,

т.е. алгоритма моделирования, и т.д. Кстати, вполне возможен обратный эффект - для сложного алгоритма процедура моделирования отрезка сигнала длительностью 0,1 с может растянуться на несколько секунд.

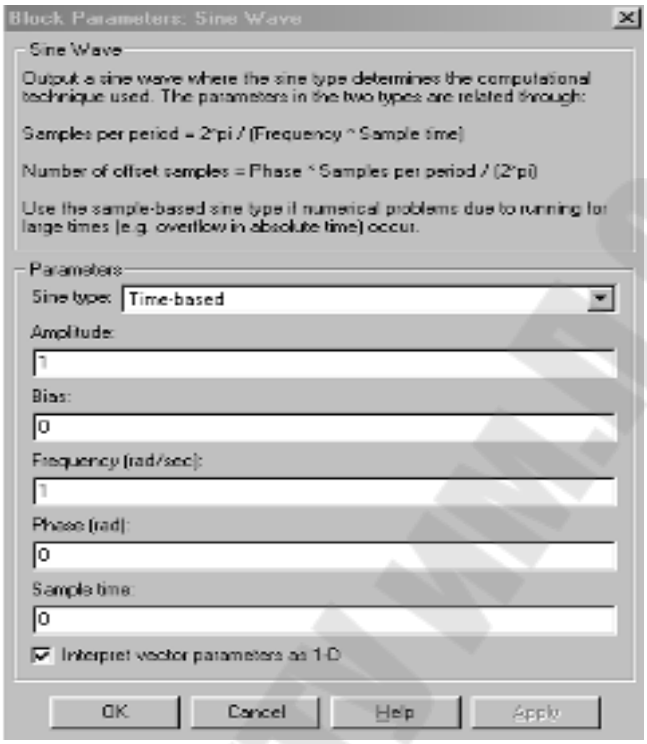


Рис. 2.6

Сигнал может генерироваться двух типов: непрерывный **time-based** и дискретный **sample-based**. Для моделирования работы непрерывных систем рекомендуют использовать непрерывный тип **time-based**, а для моделирования работы дискретных систем – дискретный тип **sample-based**.

Если установлен тип **time-based**, тогда параметр **Sample time** может принимать значения:

1) 0 (по умолчанию) – блок работает в непрерывном режиме;
2) >0 - блок работает в дискретном режиме;
3) -1 – блок наследует тот же режим, что и принимающий блок.

Как указывается в **Help**, работа в непрерывном режиме может приводить к большим погрешностям генерации на больших промежутках модельного времени.

Работа в дискретном режиме заставляет блок вести себя так, как

если бы к выходу непрерывного генератора был присоединен блок **Zero-Order Hold**. Действительно, собрав две схемы (рис. 2.7) и задав в обоих случаях значение параметра **Sample time**, равное 0,5 (окно настройки блока **Zero-Order Hold** показано на рис. 2.8), получаем идентичные результаты (рис. 2.9).

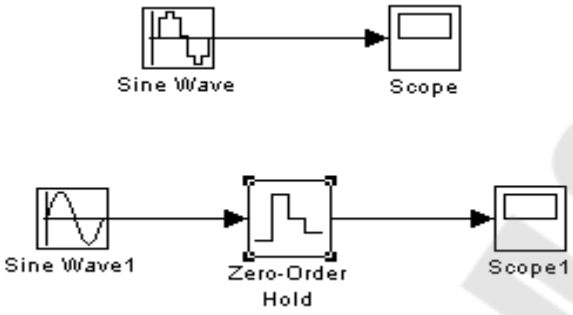


Рис. 2.7

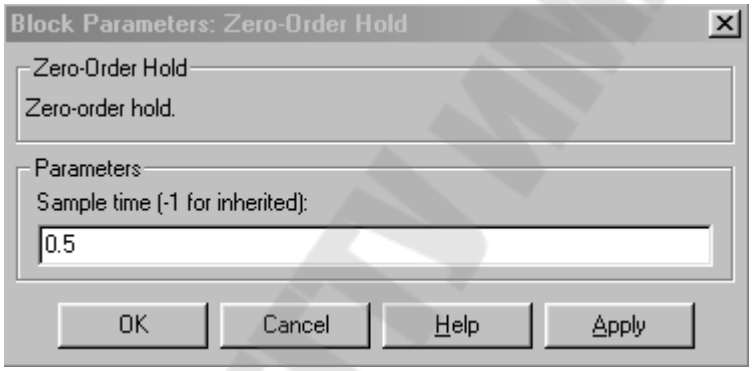


Рис. 2.8

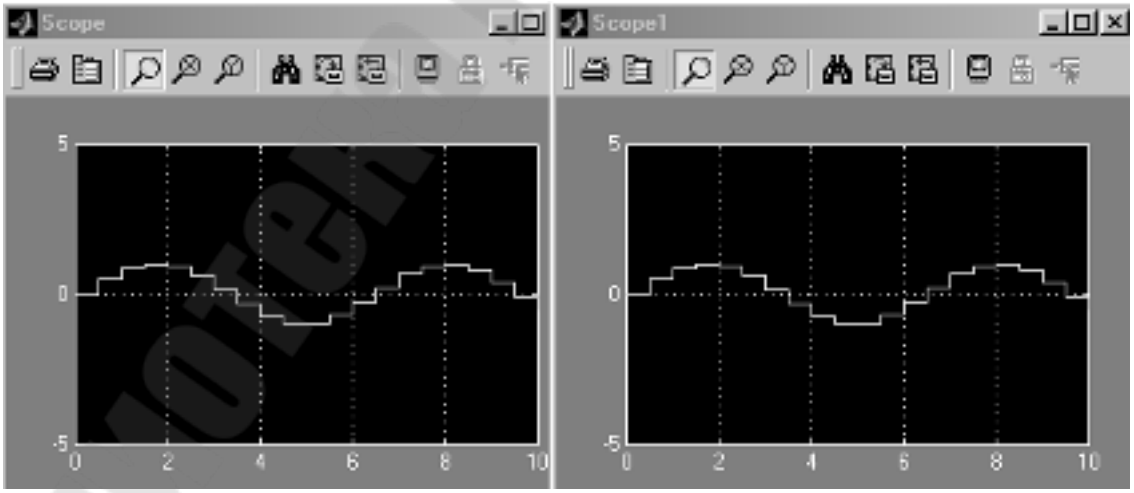


Рис. 2.9

Таким образом, блок **Zero-Order Hold** можно трактовать как “дискретизатор”, т.е. часть АЦП, ответственную за дискретизацию сигнала. Иногда блок **Zero-Order Hold** именуют АЦП . Это не сов-

сем корректно, поскольку дискретизированный сигнал в “подлинном” АЦП подвергается еще и квантованию по уровню. В блоке **Zero-Order Hold,** однако, квантование не производится.

Несколько слов о построении графиков. Помимо блока **Scope**, график можно построить и с помощью блока **X-Y-Graf**, на верхний вход X которого нужно подать последовательность моментов времени с помощью блока **Clock** (часы), а на нижний вход Y – значения генерируемого сигнала (рис. 2.10).

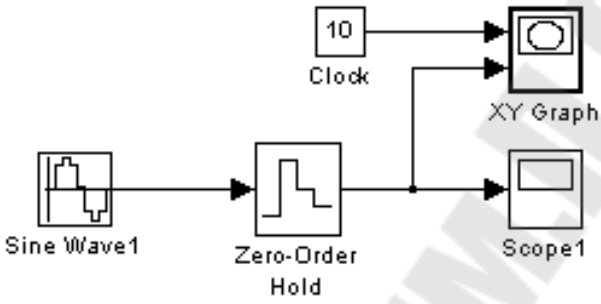


Рис. 2.10

В результате предварительно настроенный (в соответствующем окне настройки задаются граничные значения аргумента и функции, а также указывается значение параметра **Sample time**) графопостроитель выдаст показанный на рис. 2.11 график, если для блока **X-Y-Graf** задано **Sample time=-1** (т.е. период дискретизации наследуется).

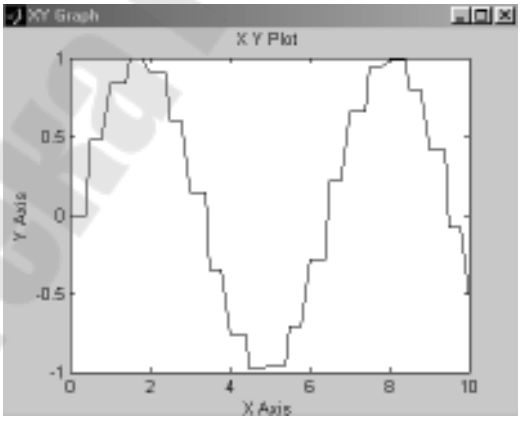


Рис. 2.11

График будет несколько иным (рис. 2.12), если для блока **X-Y-Graf** задано **Sample time=0.5**.

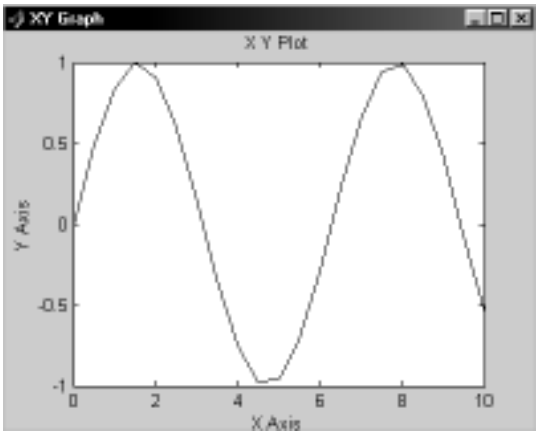


Рис. 2.12

Еще об одном способе построения графиков. Массивы отсчетов моментов времени и соответствующих значений сигнала можно с помощью блока **To Workspace** экспортировать из среды **Simulink** в среду **Matlab** (рис. 2.13).

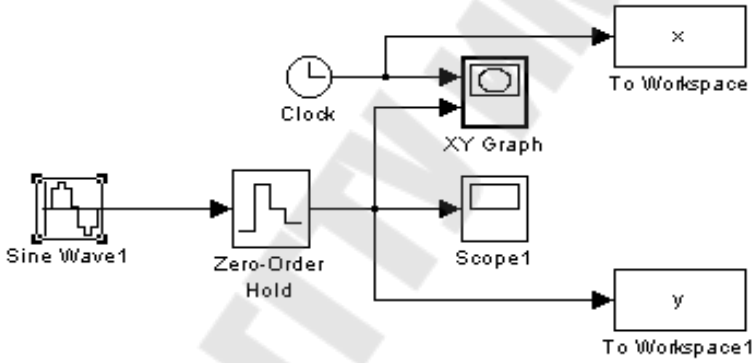


Рис. 2.13

При этом, как показывает практика, лучше всего задать формат **array** для экспортируемых данных (рис. 2.14).

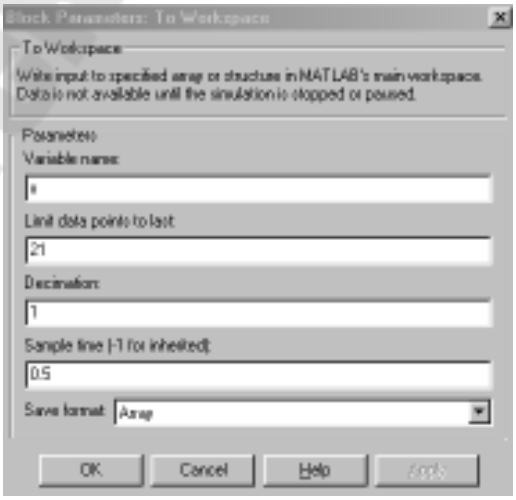


Рис. 2.14

Дальнейшее построение графика в среде **Matlab** с помощью команды **plot(x,y)** не представляет никакого труда (рис. 2.15).

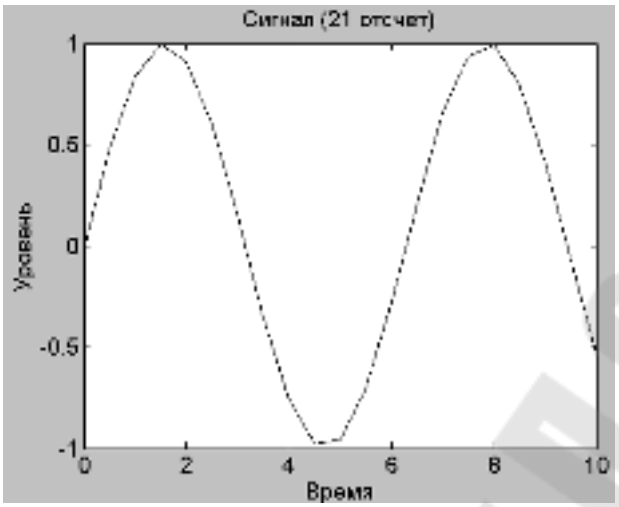


Рис. 2.15

Подытожим результаты проведенных опытов.

Сигнал типа **time-based** при работе блока генерации в режиме непрерывного времени имеет вид гладкой функции времени, а в режиме дискретного времени - вид ступенчатого сигнала, такого, как если бы к выходу генератора плавного сигнала был подсоединен блок **Zero-Order Hold,** являющийся дискретизатором типа “отсчет-хранение”.

Иными словами, задавая режим дискретного времени, мы уходим от необходимости в использовании блока **Zero-Order Hold**.

А теперь сгенерируем в **Simulink** отрезок дискретного гармонического сигнала с темы же параметрами, что были заданы в **Matlab**: амплитуда 1, частота 100 Гц, частота дискретизации 1000 Гц, начальная фаза $\pi/2$, количество отсчетов 20.

Собираем снова схему из генератора и осциллоскопа. В окне-маске настройки генератора производим указание нужных числовых значений параметров, задаем тип **time-based** и присваиваем значение **Sample time = 0.001** (рис. 2.16).

Рис. 2.16

После запуска модели получаем на экране осциллоскопа совсем не ту картину, которую ожидали (рис. 2.17).

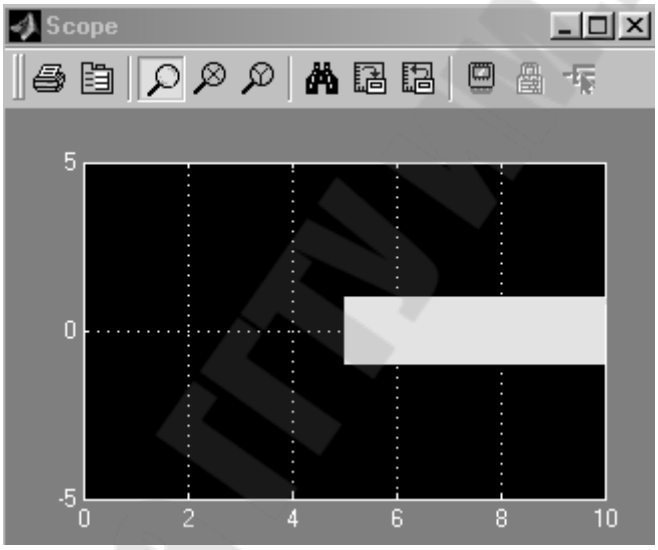
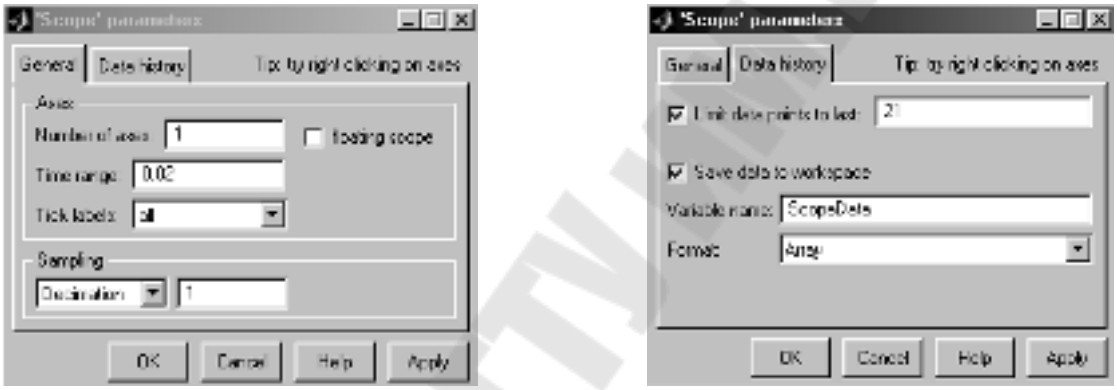


Рис. 2.17

Причина проста – нужно еще настроить параметры моделирования: задать начало и конец модельного времени (в нашем случае это 0 и 0,02 с, соответственно), а также выбрать алгоритм моделирования (тип «решателя»). На рис. 2.18 показано окно настроек параметров моделирования, активизирующееся при выборе позиции меню **Simulation/Simulation parameters**.

Рис. 2.18

Кроме того, настроим параметры осциллоскопа, щелкнув по кнопке **Parameters** на окне **Scope** (рис. 2.19a,b).



a) b)

Рис.19

После запустка модели на экране осциллоскопа появится изображение (рис. 2.20).

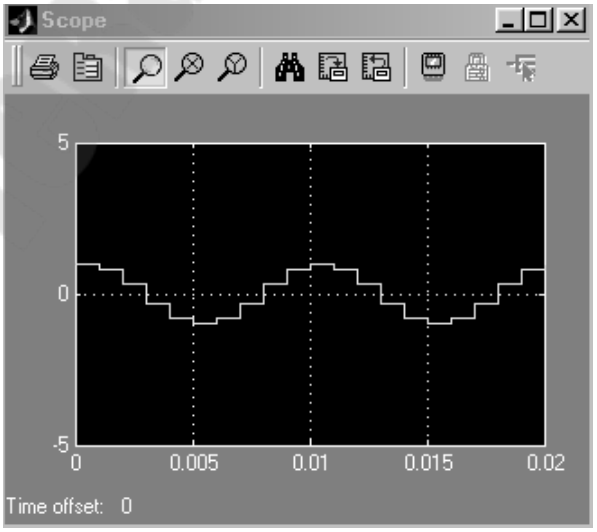


Рис. 2.20

Поскольку параметры осциллоскопа были заданы так, чтобы в рабочее пространство выводился двумерный массив **ScopeData** значений аргумента и функции, с помощью команд

```
>> y1=ScopeData(:,1);
>> y2=ScopeData(:,2);
>> plot(y1,y2)
```

можно построить график сгенерированной функции средствами **Matlab** (рис. 2.21).

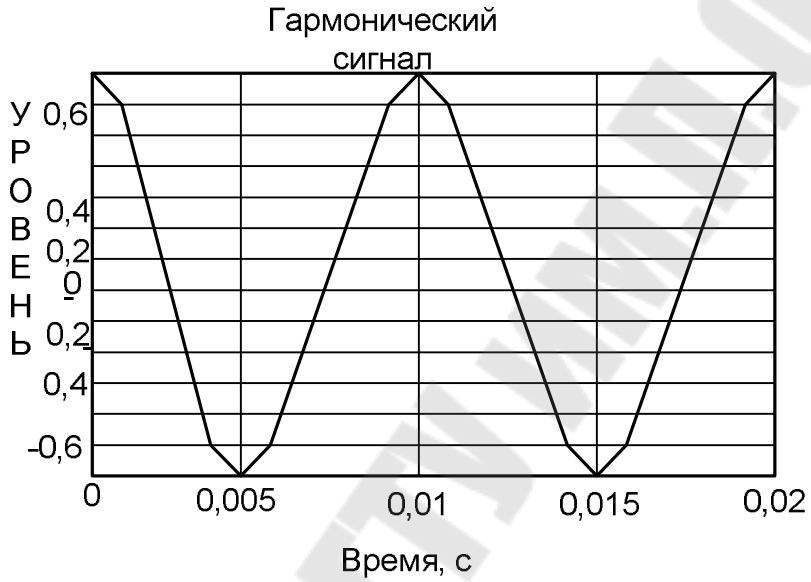


Рис. 2.21

Сравнивая рис. 2.21 и рис. 2.2, замечаем лишь одно отличие – при моделировании в **Simulink** сгенерирована 21 точка, тогда как в **Matlab** генерировалось 20 точек. Причина различия проста: на интервале модельного времени T при частоте дискретизации Fs находится $TF_s + 1$ моментов времени, для которых будет сгенерирован сигнал. Очевидно, это обстоятельство легко учесть, добившись полного совпадения результатов моделирования в средах **Matlab** и **Simulink**.

Задание для самостоятельной работы

1. Смоделировать в среде Matlab сигнал $s_2(t)$ на выходе дискретизатора (частота дискретизации Fs), если на вход дискретизатора подается сигнал:

$$s_1(t) = A_1 \cos(2\pi f_{01} t + \varphi_{01}) + A_2 \cos(2\pi f_{02} t + \varphi_{02}), \quad 0 \le t \le T .$$

Варианты значений параметров сигнала $s_1(t)$ приведены в табл.1.

Таблица 1

Вар. / Парам.	1	2	3	4	5	6	7	8
A_1	1	1	2	1	1	1	1	2
A_2	1	2	1	1	1	2	2	1
f_{01}, Гц	100	100	100	100	100	100	100	100
f_{02}, Гц	200	200	200	200	200	200	200	200
φ_{01}, рад	0	0	0	0	π	0	π	0
φ_{02}, рад	0	0	0	π	0	π	0	π

Частоту дискретизации *Fs* задать двумя способами:

а) исходя из инженерной версии теоремы Найквиста-Котельникова;

б) увеличив выбранное по п.а) значение в 5 раз.

Длительность *T* выбрать так, чтобы на ней уложилось два периода.

В отчете представить:

a) перечень команд Matlab, с помощью которых происходит вычисление сигнала $s_2(t)$;

b) график сигнала $s_2(t)$ и составляющих его гармонических компонентов.

2. Смоделировать сигнал $s_2(t)$ в среде Simulink.

В отчете представить:

c) блок-схему моделирования;

d) график сигнала $s_2(t)$ и составляющих его гармонических компонентов.

3. Сделать общие выводы по работе, сравнив между собой моделирование в среде Matlab и моделирование в среде Simulink

Контрольные вопросы:

1. Что такое «инженерная версия теоремы Найквиста-Котельникова»?
2. Что собой представляет способ дискретизации, именуемый «выборка-хранение»?
3. Как реализовать способ дискретизации, именуемый «выборка-хранение», в среде Matlab и в среде Simulink?

Литература

1. Клюев Л.Л. Теория электрической связи. Мн.:Дизайн ПРО, 1998.
2. Передача дискретных сообщений. Учебник для вузов /В.П.Шувалов, Н.В.Захарченко и др.; Под ред. В.П.Шувалова. М.: Радио и связь, 1990.
3. Основы цифровой обработки сигналов: Курс лекций / Солонина А. И. и др. – СПб.: Питер, 2005. – 768 с.: ил. 2005.
4. Сергиенко А. Б. Цифровая обработка сигналов–СПб.: Питер, 2005. – 604 с.: ил.

Содержание

Щуплов Вячеслав Валентинович

МОДЕЛИРОВАНИЕ СИГНАЛОВ И СИСТЕМ В СРЕДАХ MATLAB И SIMULINK

Лабораторный практикум по курсу «Теория электросвязи» для студентов специальности 1-36 04 02 «Промышленная электроника» дневной формы обучения В двух частях Часть 1

Подписано в печать 11.06.2010.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 1,86. Уч.-изд. л. 1,76.
Изд. № 9.
E-mail: ic@gstu.by
http://www.gstu.by

Отпечатано на цифровом дуплекаторе
с макета оригинала авторского для внутреннего использования.
Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого».
246746, г. Гомель, пр. Октября, 48.