Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет радиотехники и электроники

Кафедра информационных радиотехнологий

В. Т. Першин

ФОРМИРОВАНИЕ И ГЕНЕРИРОВАНИЕ СИГНАЛОВ ЦИФРОВОЙ РАДИОСВЯЗИ

В двух частях

Часть 1

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальностей 1-39 01 01 «Радиотехника (по направлениям)», 1-39 01 03 «Радиоинформатика»

Минск БГУИР 2016

УДК 621.391:621.396(076.5)
ББК 32.846я73+32.847я73
П27

Р е ц е н з е н т ы:

кафедра экологических информационных систем учреждения образования «Международный государственный экологический институт имени А. Д. Сахарова» Белорусского государственного университета (протокол №6 от 22.12.2014г.);

профессор кафедры радиоэлектроники филиала «Минский радиотехнический колледж» учреждения образования «Белорусский государственный университет информатики и радиоэлектроники», доктор технических наук, профессор Ф. Д. Троян

Першин, В. Т.

П27 Формирование и генерирование сигналов цифровой радиосвязи : учеб.-метод. пособие. В 2 ч. Ч. 1 / В. Т. Першин. – Минск : БГУИР, 2016. – 200 с. : ил.

ISBN 978-985-543-175-7 (ч. 1).

Учебно-методическое пособие отражает наиболее важные идеи курса «Формирование и генерирование сигналов цифровой радиосвязи», представленные в практической интерпретации в виде лабораторного практикума. Лабораторный практикум посвящен современным вопросам генерирования гармонических сигналов в схемах LC- и RC-генераторов, а также синтезаторах частоты, выполнение которых предполагается на действующих лабораторных макетах.

Учебно-методическое пособие содержит в том числе лабораторные работы по формированию сигналов с квадратурной амплитудной и фазовой манипуляцией, рассмотрены также вопросы формирования сигналов с частотной манипуляцией, которые изложены с позиций объектно-ориентированного программирования, которое нашло широкое применение разработчиками системы MATLAB.

Учебно-методическое пособие подготовлено для студентов УВО, учащихся радиотехнических колледжей и училищ.

Издано в двух частях. В части 1 приведены лабораторные работы №1–6.

УДК 621.391:621.396(076.5)
ББК 32.846я73+32.847я73

ISBN 978-985-543-175-7 (ч. 1)
ISBN 978-985-543-254-9

Предисловие

Лабораторный практикум по курсу «Формирование и генерирование сигналов цифровой радиосвязи» в настоящем издании содержит описание 14 лабораторных работ, посвященных изучению вопросов формирования сигналов с помощью LC-генераторов и RC-генераторов, анализу алгоритмов работы цифрового синтезатора частот, получения сигналов с квадратурной амплитудной и фазовой манипуляцией, сигналов с фазовой и частотной манипуляцией и сигналов с ортогональным частотным разделением с мультиплексированием (Orthogonal Frequency Bivision Multiplexing, OFDM). Рассмотрение систем связи с прямой последовательностью и частотным перескоком составляет содержание отобранных вопросов из постоянно возрастающего количества сведений, прямо или косвенно относящихся к изучаемым радиотехнологиям с расширением спектра как основным радиотехнологиям XXI века.

В качестве необходимого дополнения в содержание лабораторной работы, выполняемой с помощью пакета программ MATLAB, введены дополнительные сведения по основным положениям аппарата анализа, которым является пакет инженерных программ MATLAB, в настоящее время представляющий собой общепризнанный продукт для изучения чрезвычайно широкой номенклатуры приложений, относящихся к проблемам обработки сигналов.

Лабораторная работа №1 «Дискретные сигналы в MATLAB и SIMULINK» посвящена вопросам генерирования сигналов в MATLAB тремя способами:

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

Дискретизация аналоговых сигналов – первый шаг на пути решения задачи сопряжения аналоговых устройств и систем с дискретными сигналами.

Моделирование дискретных сигналов можно производить либо в среде MATLAB, либо в среде SIMULINK. Возможно совместное использование этих сред, что увеличивает гибкость инструментария.

Моделирование сигналов в SIMULINK удобно благодаря своей наглядности, однако требует известных навыков задания параметров блоков, из которых конструируется модель.

Важной особенностью моделирования в SIMULINK является очевидное различие понятий «реальное время» и «модельное время». Реальное время – это время, необходимое для проведения вычислений (моделирования). Модельное время – длительность моделируемого процесса.

Модельное время может быть непрерывным (time-based) и дискретным (sample-based). Непрерывное время рекомендуется использовать при моделировании непрерывных (аналоговых) систем, дискретное – дискретных (цифровых). Контроль результатов моделирования в SIMULINK можно осуществлять

как путем построения графиков, так и путем распечатки значений числовых массивов.

Построение графиков удобнее производить в среде SIMULINK. Анализ числовых данных удобнее производить, экспортируя их в рабочее пространство MATLAB.

Лабораторная работа №2 «Принципы модуляции, демодуляции и фильтрации» ставит своей целью изучить принципы модуляции, демодуляции и фильтрации с использованием пакета MATLAB. В результате выполнения этой работы студенты должны приобрести навыки применения схем беспроводной передачи информации в системах с частотным разделением каналов и написать коды системы MATLAB, нужные для приема речевых сигналов, которые модулируют несущую частоту в различных диапазонах частот. Также нужно уяснить суть частотной отсечки (frequency offset) при демодуляции сигналов.

В этой лабораторной работе изучаются технологии частотного разделения каналов (Frequency division multiplexing, FDM), которые позволяют передавать несколько информационных сигналов одновременно по одному каналу связи. FDM выделяет каждому информационному сигналу определенную частотную область внутри канала связи. Эти частотные области выбираются так, чтобы они не перекрывались.

Лабораторная работа №3 «Изучение дельта-модуляции и сигма-дельта модуляции» посвящена вопросам изучения технологий импульсной модуляции. В аналоговых системах связи помимо АМ-, ЧМ-, и ФМ- технологий используются еще технологии импульсной модуляции, такие, как Pulse Amplitude Modulation (PAM), Pulse Density Modulation (PDM), Pulse Time Modulation (PTM), Pulse Width Modulation (PWM), известная также как Pulse Duration Modulation (PDM), имеющие ряд разновидностей.

PAM – простейший вид импульсной модуляции, так как характеризуется как простотой получения модулированных колебаний, так и их детектирования. Однако этому виду модуляции присущ недостаток – он не допускает амплитудного ограничения. В связи с этим PAM малопригодна в тех случаях, когда одним из главных требований является повышение помехоустойчивости канала. Поэтому PAM находит применение в качестве промежуточного преобразования при осуществлении, а также при детектировании более сложных видов амплитудной модуляции, например, PTM или PWM. Процесс формирования сигнала амплитудной импульсной модуляции (АИМ), в котором сначала из последовательности $s_p(t)$ прямоугольных импульсов постоянной амплитуды и длительности τ, следующих с периодом повторения T, формируется последовательность $s(t)$ модулированных импульсов, которыми затем будет модулироваться несущая частота радиосигнала. При этом в качестве несущего колебания используется последовательность немодулированных импульсов, а информационным параметром является амплитуда модулированных импульсов в соответствии с передаваемым сообщением.

Используя приведенные в лабораторной работе программы, студентам предлагается изучить работу дельта-модулятора, наблюдая создаваемый им шум при изменении амплитуды входного сигнала. При этом рекомендуется

изобразить результаты работы дельта-модулятора при различных амплитудах входного сигнала, наблюдая гранулированный шум и шум, возникающий в результате перегрузки модулятора по крутизне. Перед студентами ставится задача объяснить получающиеся результаты.

Лабораторная работа №4 «Моделирование сигналов с квадратурной амплитудной и фазовой манипуляцией» посвящена вопросам моделирования сигналов с квадратурной и фазовой модуляцией. Несмотря на многообразие многопозиционных сигналов, их формирование в передатчиках и демодуляция в приемниках проводится с использованием общего технического решения, основанного на раздельном формировании двух независимых квадратурных составляющих модулирующего сигнала $I(t)$ и $Q(t)$ и их последующей передачи на одной несущей методом квадратурной амплитудной модуляции.

При выполнении деманипуляции квадратурной манипуляции демодулируемый сигнал умножается на два несущих колебания, сдвинутых по фазе друг относительно друга на 90^0, а результаты умножения пропускаются через соответствующие ФНЧ. На выходе этих фильтров получаются аналоговые сигналы синфазной и квадратурной составляющих. Далее эти сигналы дискретизируются с частотой, равной символьной скорости. Пары отсчетов синфазной и квадратурной составляющих образуют комплексное число, и ближайшая к этому числу точка используемого созвездия (а точнее – соответствующий этой точке информационный символ) выдается в качестве выходного результата.

Лабораторная работа №5 «Исследование частотно-манипулированных сигналов в системе MATLAB-SIMULINK» помимо своего непосредственного назначения охватывает более широкий круг вопросов, касающихся не только частотно-манипулированных сигналов.

Частотно-манипулированные (Frequency Shift Keying, FSK) сигналы одни из самых распространенных в современной цифровой связи. Это обусловлено прежде всего простотой их генерирования и приема ввиду нечувствительности к начальной фазе. В данной работе проводится исследование принципа формирования бинарных сигналов FSK с различными параметрами. В лабораторной работе изучается спектр FSK-сигнала. Спектр сигналов с угловой модуляцией в общем случае не выражается аналитически. Однако в случае с бинарной последовательностью можно получить оценку спектра FSK-сигналов, исходя из процедуры представления сигнала с FSK в виде суммы сигналов $s_L(t)$ и $s_H(t)$, соответствующих нижней и верхней частотам манипуляции:

$$s(t) = s_L(t)+s_H(t),\ s_L(t) = b_L(t)\cos((\omega_0-\omega_\partial)*t),\ s_H(t) = b_H(t)\cos((\omega_0-\omega_\partial)*t).$$

Таким образом, спектр FSK-сигнала $S(\omega)$ есть сумма спектров $S_L(\omega) + S_H(\omega)$ сигналов $s_L(t)$ и $s_H(t)$. Но $s_L(t)$ и $s_H(t)$ − перенесенные на соответствующие частоты сигналы $b_L(t)$ и $b_H(t)$, которые в свою очередь представляют собой последовательность импульсов длительностью $T=1/Br$.

Таким образом, видим, что составляющие FSK-сигнала разнесены на частоту девиации, а частота девиации зависит от битовой скорости Br и индекса

модуляции m. При фиксированной битовой скорости составляющие спектра FSK-сигнала будут тем ближе, чем меньше индекс FSK-модуляции.

В лабораторной работе изучаются автокорреляционные функции (АКФ) частотно-манипулированных сигналов, заданные тремя манипулирующими функциями в виде m-последовательностей, полученных с помощью четырехразрядного регистра, охваченного цепью обратной связи.

Лабораторная работа №6 «Изучение сложных методов формирования фазо- и частотно-манипулированных колебаний с использованием пакета MATLAB» посвящена вопросам формирования сигналов с фазовой и частотной манипуляцией, изложенным с позиций объектно-ориентированного программирования, которое нашло широкое применение разработчиками системы MATLAB. Поэтому здесь изложение идеи формирования сложных сигналов с фазовой и частотной манипуляцией ведется в терминах программирования объектов с помощью функций, которые достаточно подробно описаны в кратких теоретических сведениях к этой лабораторной работе. Овладение этим подходом к использованию практически неограниченных возможностей пакета MATLAB открывает необозримые возможности будущим радиоспециалистам в освоении этого продукта для практического использования в своей будущей работе.

Лабораторная работа №7 «Моделирование передающей части цифровой системы связи с OFDM средствами пакета MATLAB» и лабораторная работа №8 «Моделирование приемной части цифровой системы связи с OFDM средствами пакета MATLAB» посвящены моделированию реального сигнала с OFDM с конкретными эксплуатационными параметрами, соответствующими стандарту ETSI 300 744, относящемуся к цифровой системе передачи телевизионных сигналов. Моделирование сигнала с OFDM осуществляется применительно к формату 2k. Не все, но большая часть характерных особенностей сигнала с OFDM учтена при формировании сигнала с OFDM, так как основное внимание в этих лабораторных работах уделено именно процессу формирования сигнала с OFDM.

Лабораторная работа №9 «Генераторы гармонических колебаний» выполняется на лабораторном макете. Наиболее распространенными схемами получения гармонических колебаний являются схемы так называемых LC-генераторов с колебательным контуром для получения высокочастотных колебаний. При изучении принципов работы этих генераторов используются различные модели, анализируемые на различных уровнях теоретического рассмотрения. Краткие теоретические сведения представляют основное содержание исследования теоретических зависимостей, хорошо иллюстрируемых лабораторным наполнением практического исследования.

Автогенератор является сугубо нелинейным устройством. Тем не менее можно выделить три различных уровня, на которых рассматривается работа генератора. Этим уровням соответствуют три различные теории автогенератора, которые отличаются одна от другой предпосылками, лежащими в их основе:

- линейная теория;
- квазилинейная теория;
- нелинейная теория.

По полученным экспериментальным данным строятся графики кривой средней крутизны для жесткого и мягкого режимов работы автогенератора. Мы видим, что в жестком режиме этот график действительно имеет экстремум. Кроме того, общее поведение графика свидетельствует о значительно меньшей величине средней крутизны в жестком режиме по сравнению с величиной средней крутизны в мягком режиме. Следует обратить внимание на расчет амплитуды первой гармоники коллекторного тока в жестком режиме. В этом случае гармоническое напряжение сильнее искажается, чем в мягком режиме. Однако благодаря фильтрующему действию колебательного контура эти искажения, представляющие собой высшие гармоники базового напряжения, не будут существенным образом влиять на форму генерируемых автогенератором гармонических колебаний.

Лабораторная работа №10 «RC-генераторы» посвящена исследованию низкочастотных генераторов колебаний почти гармонической формы. Генераторы с колебательным контуром незаменимы как источники высокочастотных колебаний. Для генерирования низких частот (ниже 15…20 кГц) они неудобны, так как колебательный контур получается слишком громоздким и трудно перестраиваемым. Поэтому на этих частотах используют RC-генераторы. Лабораторная работа выполняется на лабораторных макетах и позволяет изучить целый ряд характерных особенностей их электрических принципиальных схем.

Отличие этого генератора от обычного LC-генератора заключается в том, что вместо нагрузочного колебательного контура здесь применено обычное омическое сопротивление, а обратная связь осуществляется при помощи специального четырехполюсника, составленного из конденсаторов и резисторов. Для получения устойчивой генерации на какой-либо частоте необходимо, чтобы сумма фазовых сдвигов при обходе замкнутого кольца обратной связи равнялась 2π, а коэффициент усиления транзистора являлся величиной, обратной коэффициенту обратной связи. Поскольку в схемах RC-генераторов нет избирательных цепей, то требования к выбору положения рабочей точки в таких генераторах являются очень жесткими. Положение рабочей точки в таких генераторах должно выбираться на середине линейного участка вольт-амперной характеристики (ВАХ) транзистора. Такие генераторы не могут генерировать чисто гармонические колебания, так как амплитуда генерируемых колебаний обязательно ограничивается нелинейностью ВАХ транзистора, поэтому и регулировка таких генераторов должна выполняться более тщательно: нужно внимательно следить за тем, чтобы ограничение сверху и снизу генерируемых сигналов было одинаковым.

Лабораторная работа №11 «Исследование процессов в автогенераторе на фазовой плоскости» посвящена исследованию фазовых портретов генератора с помощью пакета MathCAD или MATLAB.

Примеры выполняемых в лабораторной работе исследований показывают, что по фазовым изображениям можно судить о характере процессов, протекающих в автогенераторе. Преимущества фазового метода проявляются при исследовании систем, описываемых нелинейными дифференциальными уравнениями, т. е. когда решение в виде известных функций получено быть не может. Экспериментальное получение фазового пространства не встречает затруднений: достаточно подвести к горизонтальным пластинам осциллографа исследуемое напряжение, а к вертикальным – производную исследуемого напряжения.

Широкое использование методов компьютерной математики для решения научных и инженерных задач стимулируется не только возможностью наглядной интерпретации результатов исследования на фазовой плоскости, но и хорошо разработанными компьютерными средствами решения дифференциальных уравнений с помощью пакетов, представляющих собой системы математических вычислений, позволяющих выполнять численный анализ и символьные преобразования и выводить результаты в виде двумерных или трехмерных графиков.

Цель данной работы – получение и анализ фазовых портретов автогенератора, работающего в мягком и жестком режиме путем использования пакета компьютерной математики MathCAD или MATLAB.

Лабораторная работа №12 «Исследование синтезатора частоты» посвящена экспериментальному исследованию синтезатора частоты с прямым цифровым синтезом.

Частотный синтезатор – это устройство, с помощью которого генерируется множество дискретных частот, полученных из одного или нескольких исходных колебаний, как правило, частоты которых поддерживаются на достаточно высоком уровне стабилизации.

Структурная схема синтезатора содержит в своем составе генераторы частот одной или нескольких высокостабильных по сохранению частоты колебаний, отличающихся также высокой спектральной чистотой. В структурную схему входит также система контроля как исходных колебаний, так и колебаний, создаваемых синтезатором.

Современные синтезаторы генерируют ряд колебаний с дискретными частотами, поэтому непрактично использовать генераторы отдельных частот для каждого отдельного случая, когда требуется одна какая-то частота.

Наиболее распространенными являются следующие методы синтеза частот:

- прямой аналоговый синтез (Direct Analog Synthesis, DAS) на основе структуры смеситель – фильтр – делитель, когда частота на выходе синтезатора получается непосредственно из опорной частоты посредством операций смешения, фильтрации, умножения и деления;

- косвенный (indirect) синтез на основе фазовой подстройки частоты (Phase Locked Loop, PLL), когда выходная частота получается с помощью дополнительного генератора (чаще Voltage Controlled Oscillator, VCO), который охвачен петлей фазовой автоподстройки;

- прямой цифровой синтез (Direct Digital Synthesis, DDS), когда выходной сигнал синтезируется цифровыми методами;

- гибридный синтез, представляющий собой комбинацию нескольких методов, упомянутых выше.

Каждый из этих методов имеет свои преимущества и недостатки. Следовательно, в каждом конкретном случае нужно делать выбор, основанный на наиболее приемлемых комбинациях компромиссов.

В данной лабораторной работе изучается синтезатор частот на основе прямого цифрового синтеза.

Лабораторная работа №13 «Исследование системы с расширенным спектром прямой последовательностью» и лабораторная работа №14 «Исследование системы с расширенным спектром с перескоком частоты» посвящены исследованию современных информационных радиотехнологий систем, использующих сигналы с расширенным спектром, а также временному, частотному и кодовому разделению каналов при множественном доступе (Code Division Multiple Access, CDMA). Основное применение этих радиотехнологий – беспроводная радиосвязь, включая мобильную. В лабораторных работах рассмотрена общая структура мобильной связи, на примере которой изучаются временное, частотное и кодовое разделение каналов при множественном доступе и концепция расширения спектра сигнала, служащего средством передачи информации от одного пользователя другому.

Лабораторная работа №1
ДИСКРЕТНЫЕ СИГНАЛЫ В MATLAB И SIMULINK

1. Краткие теоретические сведения о системе MATLAB/SIMULINK

Операционная среда системы MATLAB 7 – это множество интерфейсов, которые поддерживают связь этой системы с внешним миром. Это – диалог с пользователем через командную строку или графический интерфейс, просмотр рабочей области и путей доступа, редактор и отладчик М-файлов, работа с файлами и оболочкой DOS, экспорт и импорт данных, интерактивный доступ к справочной информации, динамическое взаимодействие с внешними системами Microsoft Word, Excel и др. Реализуются эти интерфейсы через командное окно, инструментальную панель, системы просмотра рабочей области и путей доступа, редактор/отладчик М-файлов, специальные меню и т. п.

1.1. Командное окно. Инструментальная панель

Командное окно системы MATLAB содержит следующие опции, показанные на рис. 1 и описанные с помощью табл. 1.

Файлы, которые содержат коды языка MATLAB, называются М-файлами. Для создания М-файла используется текстовый редактор; вызову М-файла предшествует присваивание значений входным аргументам; результатом является значение выходной переменной. Таким образом, вся процедура включает две операции, показанные в примере 1.

***Пример* 1.** Создать М-файл, используя текстовый редактор.

Решение:

```
function c = myfile(a, b)
c = sqrt((a.^2)+(b.^2))
```

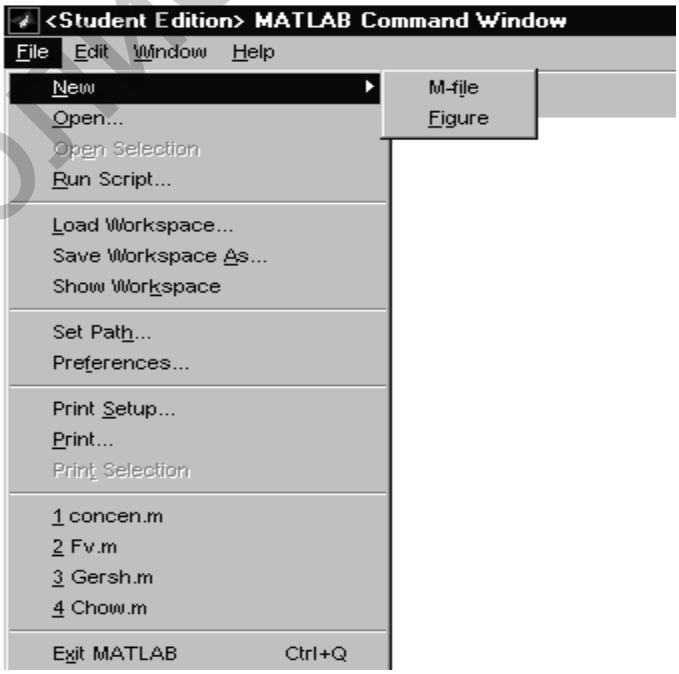


Рис. 1. Командное окно системы MATLAB

Таблица 1

Опции командного окна

Опция	*Подопции*	*Назначение*
New	M-file Figure	Открыть в редакторе/отладчике новый файл Открыть графическое окно
Open		Открыть в редакторе/отладчике указанный файл
Open Selection		Открыть в редакторе/отладчике файл, выделенный в произвольной строке командного окна
Run Script		Вызов окна для запуска Script-файла
Load Workspace		Вызов окна загрузки MAT-файла
Save Workspace As		Вызов окна сохранения MAT-файла
Show Workspace		Вызов средства просмотра рабочей области Workspace Browser
Set Path		Вызов средства просмотра путей доступа Path Browser
Preferences		Выбор характеристик
Print Setup		Установка опций принтера
Print		Установка опций вывода на печать
Print Selection		Печать выделенного фрагмента

***Пример* 2.** Ввести численные значения переменных *a* и *b* и вычислить значение переменной *c*, используя М-файл, созданный в примере 1.

Решение: Введем значения переменных *a* = 7.5 и *b* = 3.342 и затем вызываем М-файл из командной строки. Покажем, как это делается:

```
>> a = 7.5
>> b = 3.342
>> c = myfile(a, b)
>> ans
        c = 8.2109
```

1.2. Типы М-файлов

Существует два типа М-файлов: М-сценарии и М-функции со следующими характеристиками, описание которых представлено в табл. 2.

Таблица 2

Характеристики М-сценариев и М-функций

М-сценарий	М-функция
Не использует входных и выходных аргументов	Использует входные и выходные аргументы
Оперирует с данными из рабочей области	По умолчанию внутренние переменные являются локальными по отношению к функции
Предназначен для автоматизации последовательности шагов, которые нужно выполнять много раз	Предназначена для расширения возможностей языка MATLAB (библиотеки функций, пакеты прикладных программ)

1.3. Структура М-файла

М-файл, оформленный в виде функции, состоит из следующих компонентов:

- строка определения функции, function f = fact (n);
- первая строка комментария, %fact, т. е. вычисление факториала;
- комментарий, %fact(n) возвращает n! – факториал числа n;
- тело функции, %вычислить fact(n) = prod(1:n) f = prod(1:n).

Структура этой простейшей функции содержит компоненты, которые являются общими для любых функций системы MATLAB:

Строка определения функции задает имя, количество и порядок следования входных и выходных аргументов.

Первая строка комментария определяет назначение функции. Она выводится на экран с помощью команд lookfor или help и имя каталога.

Комментарий выводится на экран вместе с первой строкой при использовании команды help и имя функции.

Тело функции – это программный код, который реализует вычисления и присваивает значения выходным аргументам.

М-функцию можно вызвать из командной строки системы MATLAB или из других М-файлов, обязательно указав все необходимые атрибуты – входные аргументы в круглых скобках, выходные аргументы в квадратных скобках.

Вызов функции. При вызове М-функции система MATLAB транслирует функцию в псевдокод и загружает в память. Это позволяет избежать повторного синтаксического анализа. Псевдокод остается в памяти до тех пор, пока не будет использована команда clear или завершен сеанс работы.

Допустимы следующие модификации команды clear, описание которых приведено в табл. 3.

Таблица 3

Модификации команды clear

clear <имя_функции>	Удалить указанную функцию из рабочей области
clear functions	Удалить все откомпилированные программы
clear all	Удалить программы и данные

1.4. Типы переменных

Локальные и глобальные переменные. Использование переменных в М-файле ничем не отличается от использования переменных в командной строке, а именно:

- переменные не требуют объявления, но прежде чем переменной присвоить значение, необходимо убедиться, что всем переменным в правой части значения присвоены;
- любая операция присваивания создает переменную, если это необходимо, или изменяет значение существующей переменной;
- имена переменных начинаются с буквы, за которой следует любое количество букв, цифр и подчеркиваний (система MATLAB различает символы верхнего и нижнего регистров);
- имя переменной не должно превышать длину в 31 символ (более точно, имя может быть и длиннее, но система MATLAB принимает во внимание только первый 31 символ).

Обычно каждая М-функция, задаваемая в виде М-файла, имеет собственные локальные переменные, которые отличны от переменных других функций и переменных рабочей области. Однако, если несколько функций и рабочая область объявляют некоторую переменную глобальной, то все они используют единственную копию этой переменной. Любое присваивание этой переменной распространяется на все функции, где она объявлена глобальной.

***Пример* 3**. Допустим, требуется исследовать влияние коэффициентов *a* и *b* для модели хищник – жертва, описываемой уравнениями Лотке – Вольтерра.

Решение:

Создадим М-файл lotka.m:

```
function yp = lotka(t, y) %LOTKA уравнения Лотке – Вольтерра для модели хищник – жертва
global ALPHA BETA yp = [y(1) - ALPHA*y(1)*y(2); -y(2) + BETA*y(1)*y(2)];
```

Затем через командную строку введем операторы:

```
global ALPHA BETA
ALPHA = 0.01;
BETA = 0.02;
[t,y] = ode23('lotka2',[0 10],[1; 1]);
plot(t,y)
```

Команда global объявляет переменные ALPHA и BETA глобальными и, следовательно, доступными в функции lotka.m. Таким образом, они могут быть изменены из командной строки, а новые решения будут получены без редактирования М-файла lotka.m.

Для работы с глобальными переменными необходимо:

- объявить переменную как глобальную в каждой М-функции, которая необходима эта переменная (для того чтобы переменная рабочей области была глобальной, необходимо объявить ее как глобальную из командной строки);
- в каждой функции использовать команду global перед первым появлением переменной (рекомендуется указывать команду global в начале М-файла).

Имена глобальных переменных обычно более длинные и более содержательные, чем имена локальных переменных, и часто используют заглавные буквы. Это необязательно, но рекомендуется, чтобы обеспечить удобочитаемость кода языка MATLAB и уменьшить вероятность случайного переопределения глобальной переменной.

Специальные переменные. Некоторые М-функции возвращают специальные переменные, характеристики которых приведены в табл. 4. Специальные переменные играют важную роль при работе в среде системы MATLAB.

Таблица 4

Характеристики специальных переменных

ans	Последний результат, если выходная переменная не указана, то MATLAB использует переменную ans
eps	Точность вычислений с плавающей точкой определяется длиной мантиссы и для PC eps = 2.220446049250313e-016
realmax	Максимальное число с плавающей точкой, представимое в компьютере; для PC realmax = 1.797693134862316e+308
realmin	Минимальное число с плавающей точкой, представимое в компьютере; для PC realmin = 2.225073858507202e-308
pi	Специальная переменная для числа p: pi=3.141592653589793e+000
i, j	Специальные переменные для обозначения мнимой единицы

1.5. Создание М-файлов: М-сценарии и М-функции

М-файлы являются обычными текстовыми файлами, которые создаются с помощью текстового редактора. Для операционной среды персонального компьютера система MATLAB поддерживает специальный встроенный редактор/отладчик, хотя можно использовать и любой другой текстовый редактор с ASCII-кодами.

Открыть редактор можно двумя способами:

- из меню File выбрать опцию New, а затем M-File;
- использовать команду редактирования edit.

***Пример* 4.** Команда edit poof запускает редактор и открывает файл poof.m. Если имя файла опущено, то запускается редактор и открывается файл без имени. Записать функцию fact, вводя строки текста и сохраняя их в файле с именем fact.m в текущем каталоге, и вычислить ее значение при переменной, равной 5.

Решение:

Как только такой файл создан, можно выполнить следующие команды:

- вывести на экран имена файлов текущего каталога;
- набрать слово **what**;
- вывести на экран текст М-файла fact.m, напечатав в командном окне: **type fact**;
- вызвать функцию fact с заданными параметрами, т. е. со значением переменной, равным 5.

Имеем

```
>> fact (5) ans  120
```

М-сценарии являются самым простым типом М-файла – у них нет входных и выходных аргументов. Они используются для автоматизации многократно выполняемых вычислений. Сценарии оперируют данными из рабочей области и могут генерировать новые данные для последующей обработки в этом же файле. Данные, которые используются в сценарии, сохраняются в рабочей области после завершения сценария и могут быть использованы для дальнейших вычислений. Структура М-сценария представлена в табл. 5 применительно к решению задачи, поставленной в условии примера 5.

***Пример* 5**. Вычислить радиус-вектор rho для различных тригонометрических функций в зависимости от угла theta и построить последовательность графиков в полярных координатах.

Таблица 5

Структура М-сценария

Строка комментария	% M-file petals – сценарий построения лепесткового графика

Вычисления	theta = -pi:0.01:pi;
	rho(1, :) = 2*sin(5*theta).^2;
	rho(2, :) = cos(10*theta).^3;
	rho(3, :) = sin(theta).^2;
	rho(4, :) = 5*cos(3.5*theta).^3;

Команды графического вывода	for i = 1:4 polar (theta, rho(i, :)) pause end *(не приведены)*

Создайте М-файл petals.m, вводя указанные выше операторы. Этот файл является сценарием. Ввод команды petals.m в командной строке системы MATLAB вызывает выполнение операторов этого сценария.

После того как сценарий отобразит первый график, нажмите клавишу Return, чтобы перейти к следующему графику. В сценарии отсутствуют входные и выходные аргументы; программа petals.m сама создает переменные, которые сохраняются в рабочей области системы MATLAB. Когда выполнение завершено, переменные (i, theta и rho) остаются в рабочей области. Для того чтобы увидеть этот список, следует воспользоваться командой whos.

М-функции являются М-файлами, которые допускают наличие входных и выходных аргументов. Они работают с переменными в пределах собственной рабочей области, отличной от рабочей области системы MATLAB.

***Пример* 6.** Вычислить среднее значение элементов вектора.

Решение:

Функция average – это достаточно простой М-файл, который вычисляет среднее значение элементов вектора:

```
function y = average (x) % AVERAGE Среднее значение элементов вектора.
% AVERAGE(X), где X – вектор. Вычисляет среднее значение элементов вектора.
% Если входной аргумент не является вектором, генерируется ошибка
[m,n] = size(x);
if (~((m == 1) | (n == 1)) | (m == 1 & n == 1)) error('Входной массив должен быть вектором') end y =sum(x)/length(x);
% Собственно вычисление.
```

Попробуйте ввести эти команды в М-файл, именуемый average.m. Функция average допускает единственный входной и единственный выходной аргументы. Для того чтобы вызвать функцию average, надо ввести следующие операторы:

```
z = 1:99; average(z) ans = 50
```

1.6. Структура М-функции

М-функция состоит из:

- строки определения функции;

- первой строки комментария;
- собственно комментария;
- тела функции;
- строчных комментариев.

Строка определения функции. Строка определения функции сообщает системе MATLAB, что файл является М-функцией, а также определяет список входных аргументов.

***Пример* 7.** Записать строку определения функции average.

Решение:

Строка определения функции average имеет вид

```
function y = average(x)
```

Здесь:

function – ключевое слово, определяющее М-функцию;

y – выходной аргумент;

average – имя функции;

x – входной аргумент.

Каждая функция в системе MATLAB содержит строку определения функции, подобную приведенной. Если функция имеет более одного выходного аргумента, список выходных аргументов помещается в квадратные скобки. Входные аргументы, если они присутствуют, помещаются в круглые скобки. Для отделения аргументов во входном и выходном списках применяются запятые.

Имена входных переменных могут, но не обязаны совпадать с именами, указанными в строке определения функции.

Первая строка комментария. Для функции average первая строка комментария выглядит так:

```
% AVERAGE Среднее значение элементов вектора
```

Это – первая строка текста, которая появляется, когда пользователь набирает команду help <имя_функции>. Кроме того, первая строка комментария выводится на экран по команде поиска lookfor. Поскольку эта строка содержит важную информацию об М-файле, она должна быть тщательно составлена.

Комментарий. Для М-файлов можно создать online-подсказку, вводя текст в одной или более строках комментария.

***Пример* 8.** Сформировать текст комментария функции.

Решение:

Сформируем несколько строк комментария:

```
% Функция average(x) вычисляет среднее значение элементов вектора x.
% Если входной аргумент не является вектором, выдается ошибка.
```

При вводе команды подсказки help <имя_функции> система MATLAB отображает строки комментария, которые размещаются между строкой определения функции и первой пустой строкой, либо началом программы. Команда help <имя_функции> игнорирует комментарии, размещенные вне этой области.

***Пример* 9.** Отобразить комментарий к функции sin(x).

Решение:

```
help sin SIN Sine.
```

Функция *sin(x)* is the sine of the elements of *x*. Функция *sin(x)* вычисляет функцию синуса элементов массива *x*.

Оглавление каталога. Можно создать комментарий для целого каталога, если сформировать специальный файл с именем Contents.m. Этот файл должен содержать только строки комментариев.

MATLAB выводит на экран строки файла Contents.m по команде help <имя_каталога>.

Если каталог не содержит файла Contents.m, то по команде help <имя_каталога> распечатывается первая строка комментария для каждого М-файла данного каталога.

Тело функции. Тело функции содержит код языка MATLAB, который выполняет вычисления и присваивает значения выходным аргументам. Операторы в теле функции могут состоять из вызовов функций, программных конструкций для управления потоком команд, интерактивного ввода/вывода, вычислений, присваиваний, комментариев и пустых строк.

Как уже говорилось ранее, комментарии отмечаются знаком (%). Строка комментария может быть размещена в любом месте М-файла, в том числе и в конце строки.

Кроме строк комментариев в текст М-файла можно включать пустые строки. Однако надо помнить, что пустая строка может служить указателем окончания подсказки.

Имена М-функций. В системе MATLAB на имена М-функций налагаются те же ограничения, что и на имена переменных – их длина не должна превышать 31 символа. Более точно, имя может быть и длиннее, но система MATLAB принимает во внимание только первый 31 символ. Имена М-функций должны начинаться с буквы; остальные символы могут быть любой комбинацией букв, цифр и знака подчеркивания.

Имя файла, содержащего М-функцию, составляется из имени функции и расширения ".m".

Если имя файла и имя функции в строке определения функции разные, то используется имя файла, а внутреннее имя игнорируется.

Хотя имя функции, определенное в строке определения функции, может и не совпадать с именем файла, настоятельно рекомендуется использовать одинаковые имена.

Двойственность функций и команд. Команды системы MATLAB – это операторы вида load help.

Многие команды могут быть модифицированы добавлением операндов load August17.dat help magic type rank.

Альтернативный метод задания модификаторов – определить их в качестве строковых аргументов функции:

load('August17.dat') help('magic') type('rank')

В этом заключается двойственность понятий команды и функции в системе MATLAB. Любая команда вида

command argument

может быть записана в форме функции command('argument').

Преимущество функционального описания проявляется, когда строка аргументов формируется по частям.

1.7. Типы данных

Типы данных системы MATLAB можно описать с помощью табл. 6.

Таблица 6

Типы данных системы MATLAB

Класс	*Пример*	*Описание*
Double	[1 2; 3 4] 5 + 6i	**Числовой массив** удвоенной точности (это наиболее распространенный тип переменной в системе MATLAB
Char	'Привет'	**Массив символов** (каждый символ – длиной 16 бит), часто именуется строкой
Sparse	Speye(5)	**Разреженная матрица** удвоенной точности (только двумерная). Разреженная структура применяется для хранения матриц с небольшим количеством ненулевых элементов, что позволяет использовать лишь небольшую часть памяти, требуемой для хранения полной матрицы. Разреженные матрицы требуют применения специальных методов для решения задач
Cell	{ 17 'привет' eye (2)}	**Массив ячеек**. Элементы этого массива содержат другие массивы. Массивы ячеек позволяют объединить связанные данные, возможно различных размеров, в единую структуру
Struct	A.day = 12; A.color = 'Red'; A.mat = magic(3);	**Массив записей**. Он включает имена полей. Поля сами могут содержать массивы. Подобно массивам ячеек массивы записей объединяют связанные данные и информацию о них
Uint8	Uint8 (magic (3))	**Массив 8-разрядных целых чисел без знаков**. Он позволяет хранить целые числа в диапазоне от 0 до 255 в 1/8 части памяти, требуемой для массива удвоенной точности. Никакие математические операции для этих массивов не определены
UserObject	inline('sin(x)')	Тип данных, определяемый пользователем

2. Выполнение работы

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. Сигнал, изображенный на рис. 2, а, будем называть *исходным аналоговым.* На рис. 2, 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. 2, 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, в и рис. 2, 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, г отличаются от импульсов рис. 2, в тем, что полностью заполняют промежуток между моментами обновления значения сигнала. Такой способ дискретизации, именуемый «выборка – хранение», наиболее эффективен с точки зрения помехоустойчивости.

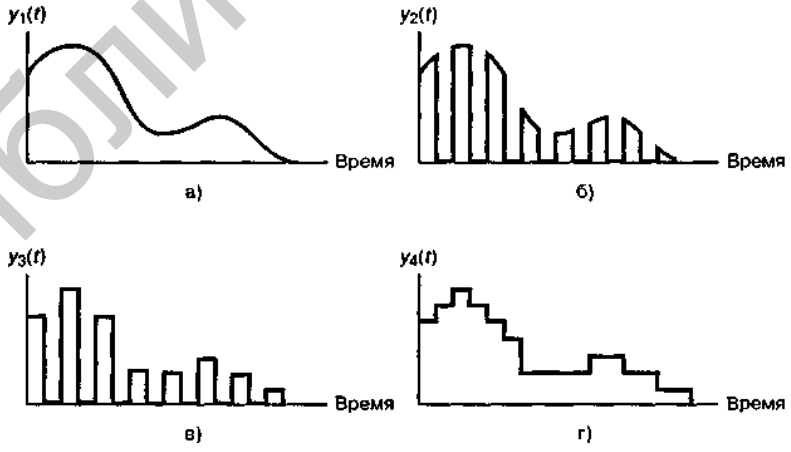


Рис. 2. Исходные данные в системе координат «время – амплитуда»:
а – исходный аналоговый сигнал; б – данные в естественной дискретизации;
в – квантованные выборки; г – выборка – хранение

2.1. Моделирование дискретных сигналов в MATLAB

Генерировать сигналы в MATLAB можно тремя способами:

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

2.1.1. 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, c'); 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. 3).

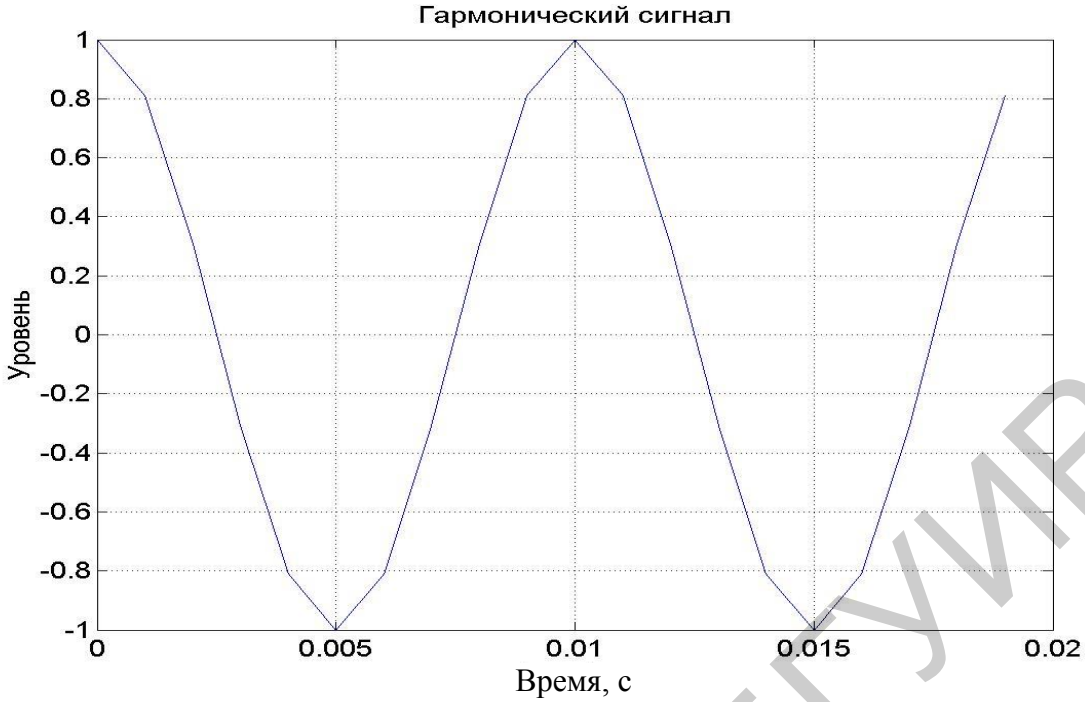


Рис. 3. Генерирование гармонического сигнала в диалоговом режиме

Примечание. При использовании символов кириллицы в тексте команд следует учитывать особенности отношения каждой конкретной версии MATLAB к кириллице.

2.1.2. Генерирование сигналов путем создания m-скрипта

Данный способ отличается тем, что все команды набираются в специальном окне редактора m-файлов (рис. 4).

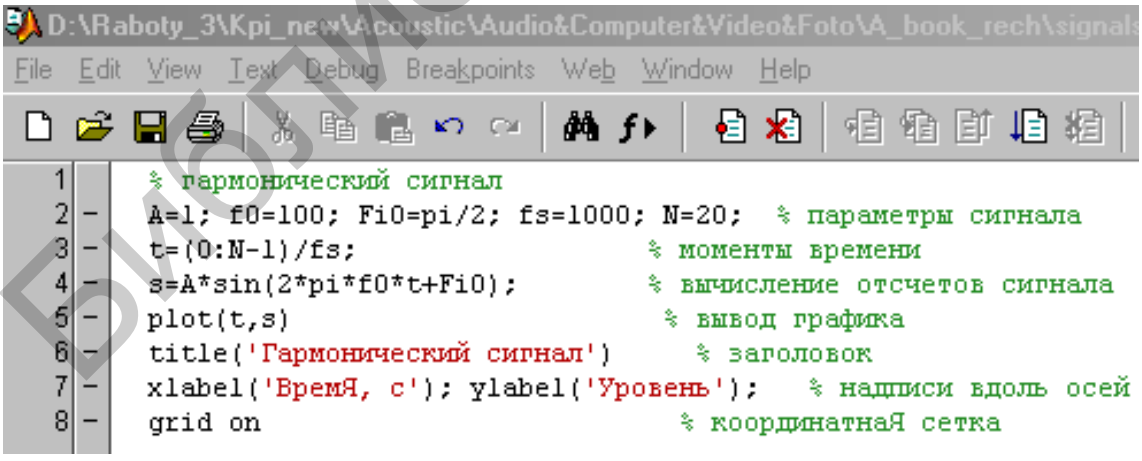


Рис. 4. Генерирование сигналов путем создания m-скрипта

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

2.1.3. Генерирование сигналов путем создания m-функции

Данный способ отличается тем, что входные данные записываются, 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.*

2.2. Моделирование дискретных сигналов в SIMULINK

Генерирование сигналов в SIMULINK 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. 5).

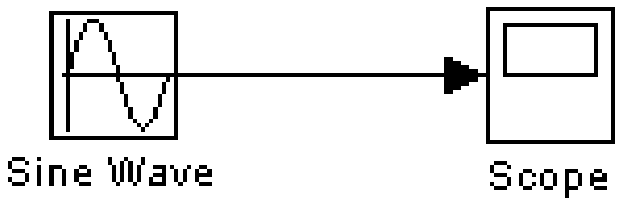


Рис. 5. Структурная схема генерирования гармонического сигнала в SIMULINK

Затем двойным щелчком по блоку осциллоскопа активизируем окно, имитирующее экран осциллоскопа, и запустим модель (кнопка **Start simulation**). В результате получим изображение отрезка синусоиды (рис. 6).

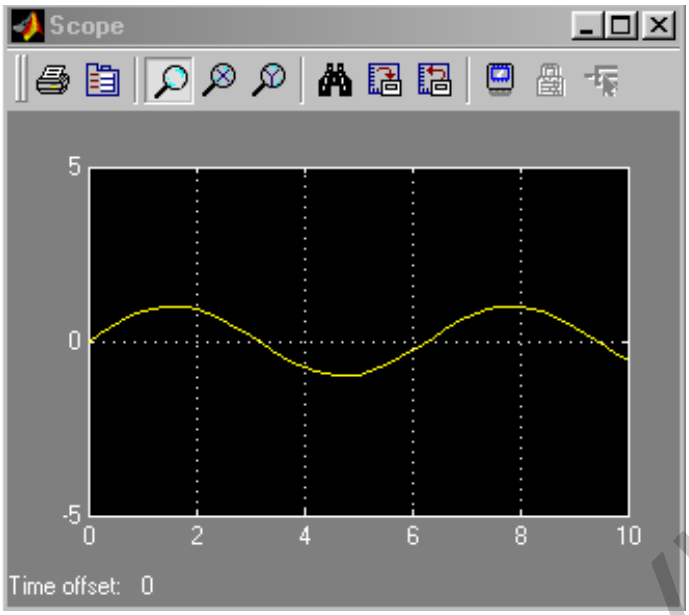


Рис. 6. Моделирование дискретного сигнала в SIMULINK

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. 9). 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. 9 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.

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:

- 0 (по умолчанию) – блок работает в непрерывном режиме;
- >0 – блок работает в дискретном режиме;
- 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. 8) и задав в обоих случаях значение параметра **Sample time**, равное 0.5 (окно настройки блока **Sine Wave** показано на рис. 9 и окно настройки блока **Zero-Order Hold** показано на рис. 10), получаем идентичные результаты (рис. 11).

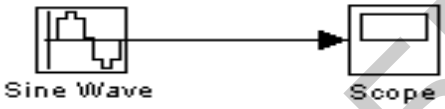


Рис. 7. Схема генерирования сигнала в SIMULINK

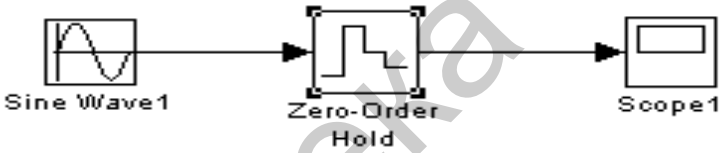


Рис. 8. Структурные схемы генерирования гармонического сигнала

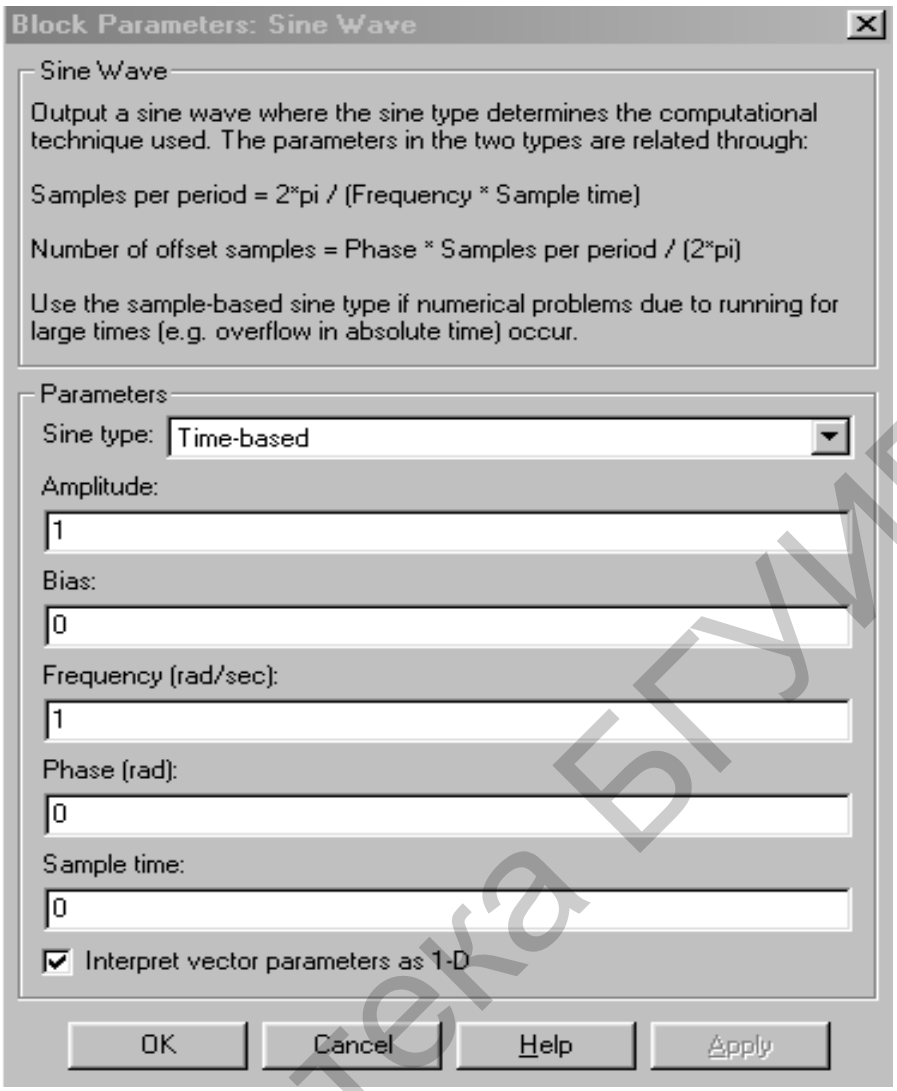


Рис. 9. Окно настройки блока Sine Wave

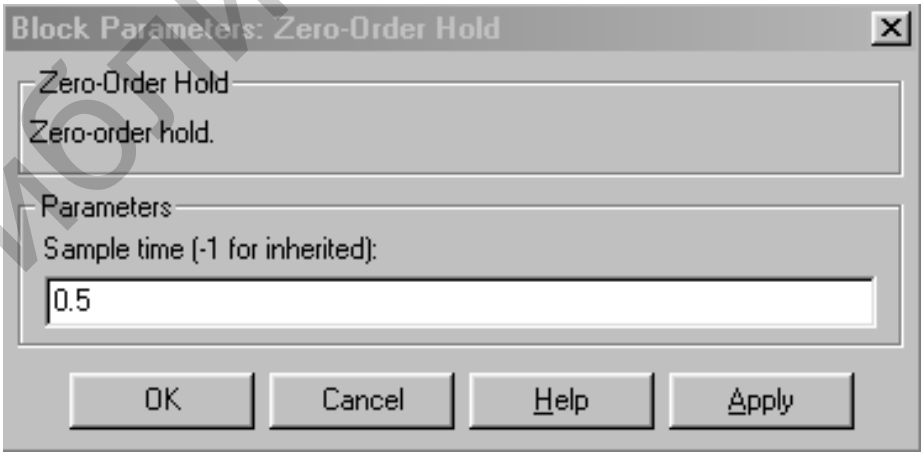


Рис. 10. Окно настройки блока Zero-Order Hold

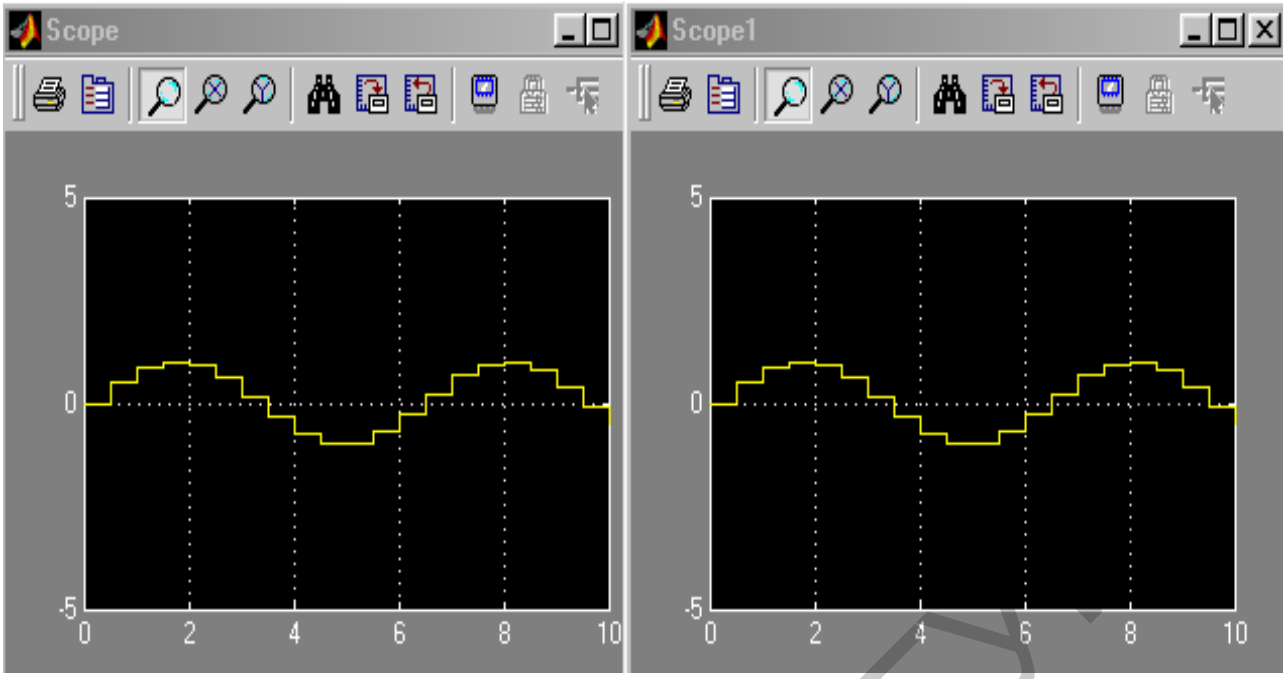


Рис. 11. Результаты моделирования гармонического сигнала

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. По нашему мнению, это не 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.12).

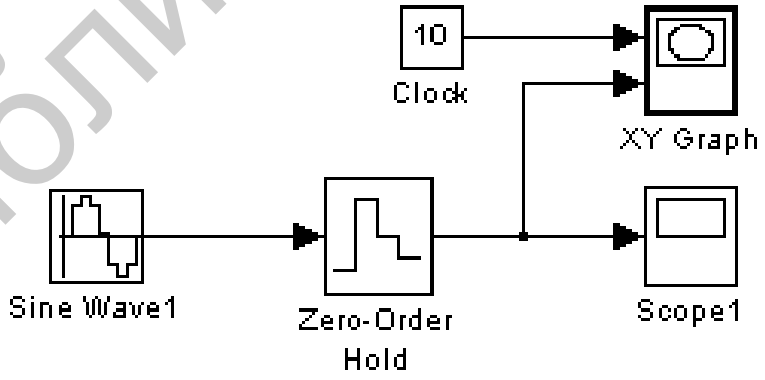


Рис. 12. Структурная схема использования блока X-Y-Graf

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.13 график, если для блока **X-Y-Graf** задано **Sample time=-1** (т. е. период дискретизации наследуется).

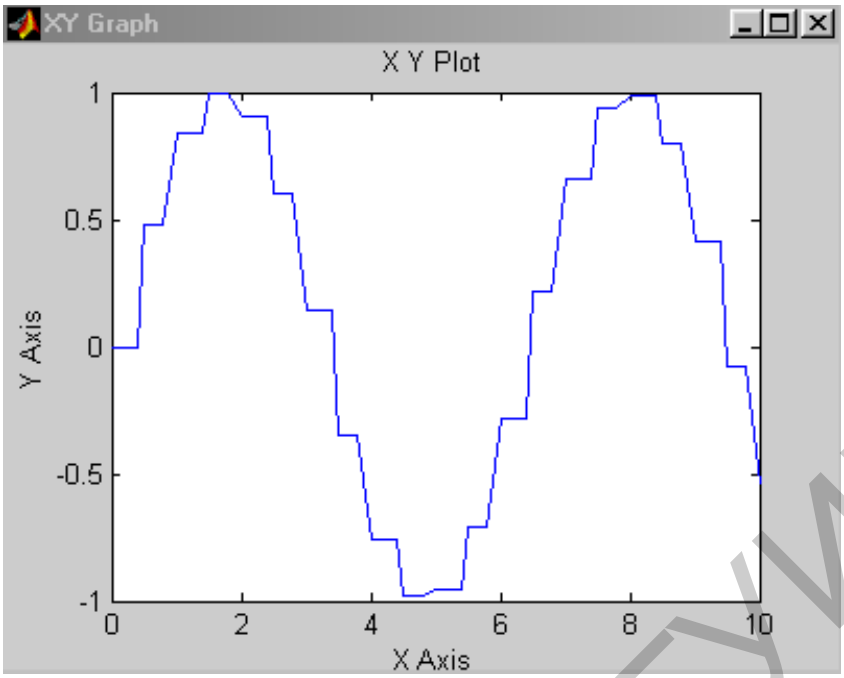


Рис. 13. Вид графика, когда период дискретизации не наследуется

График будет несколько иным (рис. 14), если для блока **X-Y-Graf** задано **Sample time=0.5**.

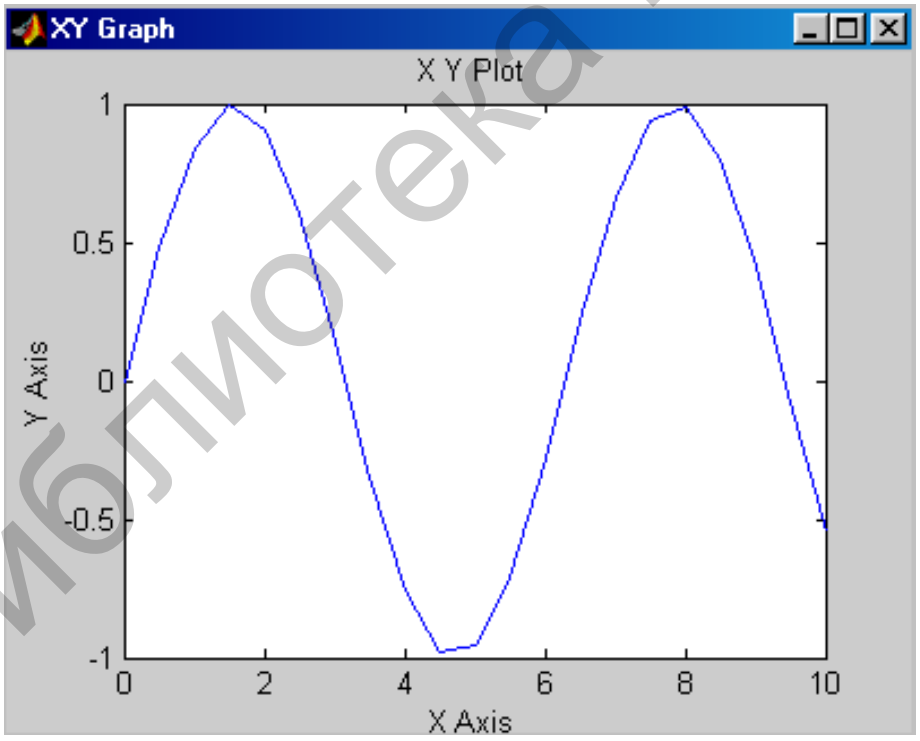


Рис. 14. Вид графика, когда задано Sample time = 0.5

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. 15).

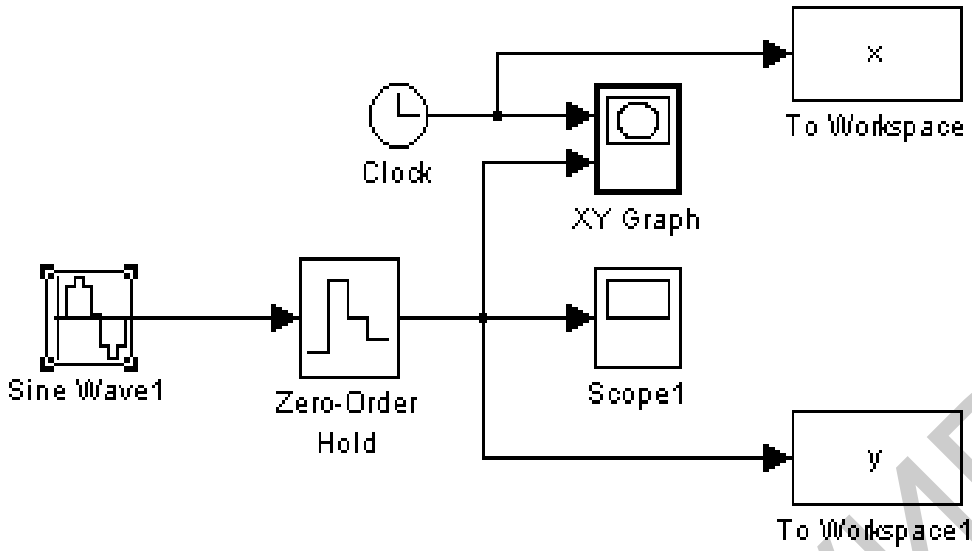


Рис. 15. Структурная схема экспортирования данных из среды MATLAB в среду SIMULINK

При этом лучше всего задать формат **array** для экспортируемых данных (рис. 16).

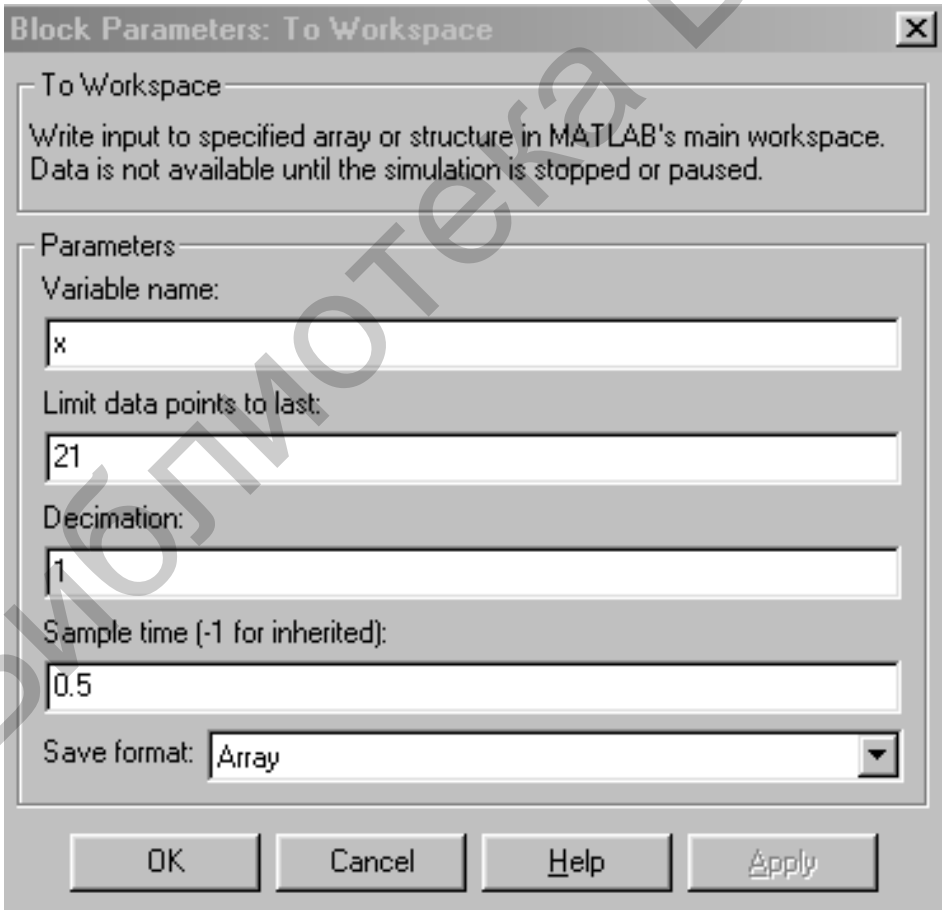


Рис. 16. Задание формата array для экспортируемых данных

Дальнейшее построение графика в среде MATLAB с помощью команды **plot(x,y)** не представляет никакого труда (рис. 17).

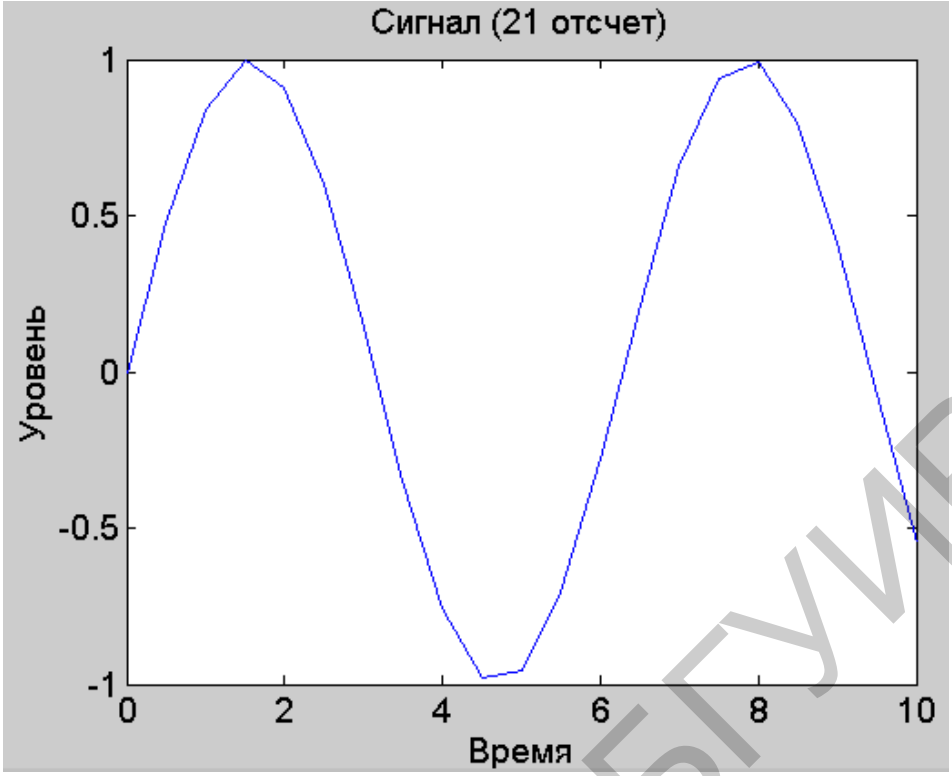


Рис. 17. Построение графика в среде MATLAB с помощью команды plot(x,y)

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 использования блока **Zero-Order Hold**.

А теперь сгенерируем в SIMULINK отрезок дискретного гармонического сигнала с теми 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. 18).

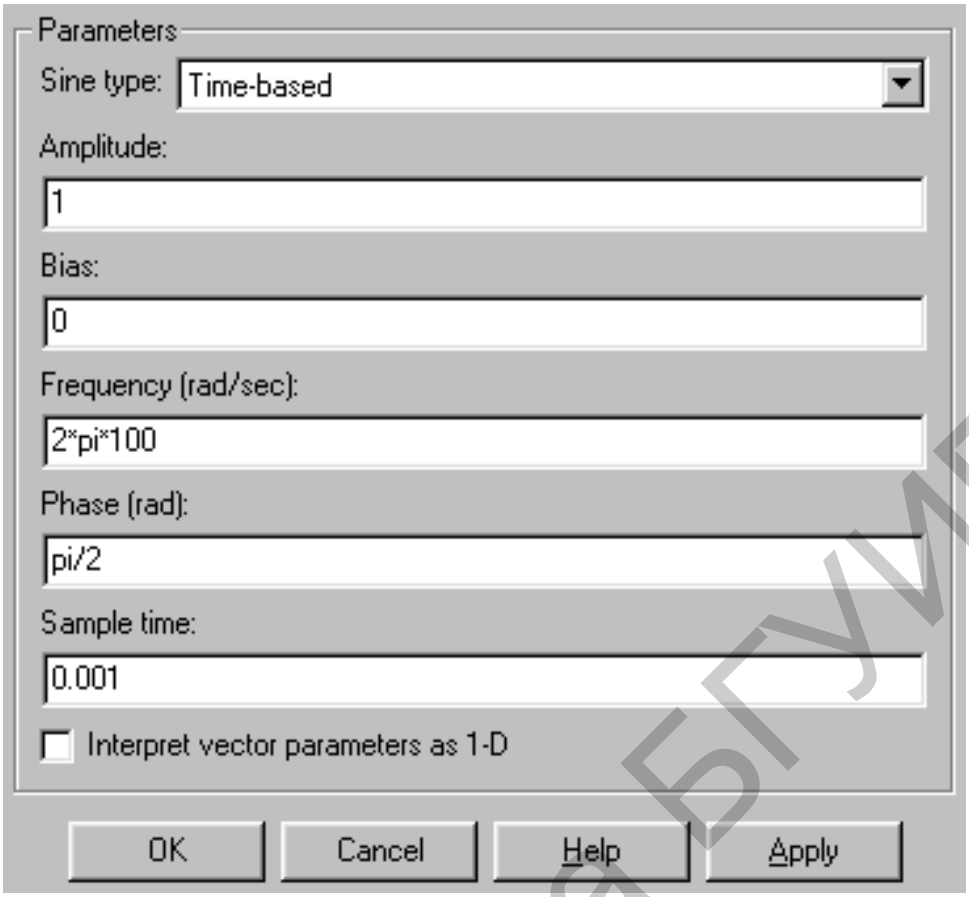


Рис. 18. Окно для выбора значений параметров генератора сигнала

После запуска модели получаем на экране осциллоскопа совсем не ту картину, которую ожидали (рис. 19).

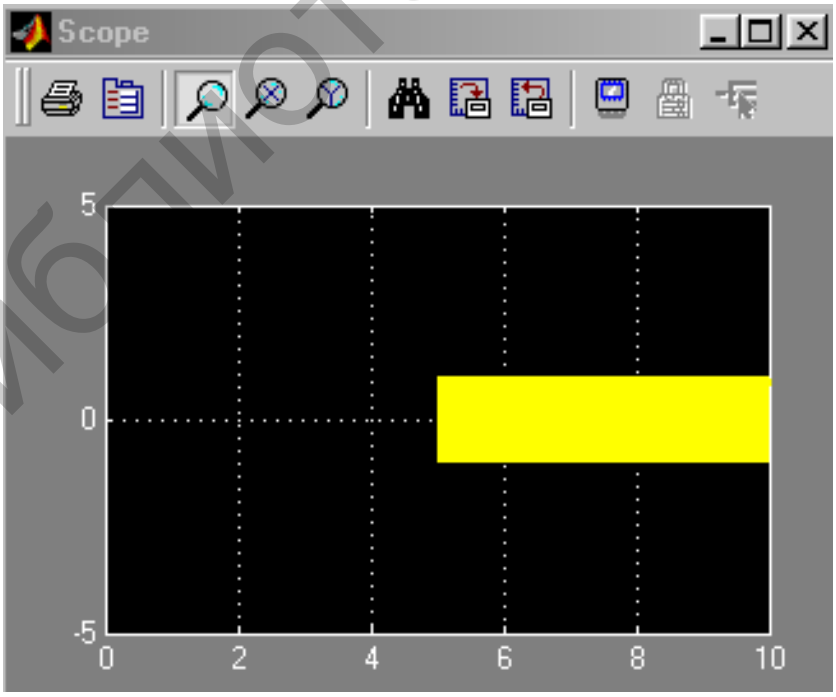


Рис. 19. Экран осциллоскопа при неправильно установленных параметрах генератора

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. 20 показано окно настроек параметров моделирования, активизирующееся при выборе позиции меню **Simulation/Simulation parameters**.

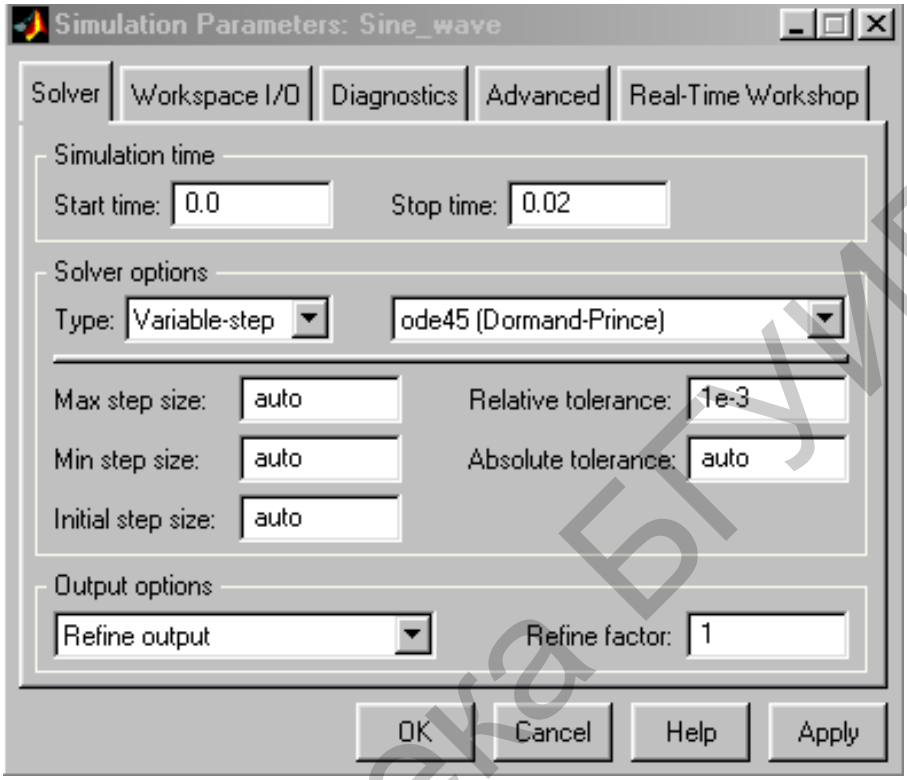
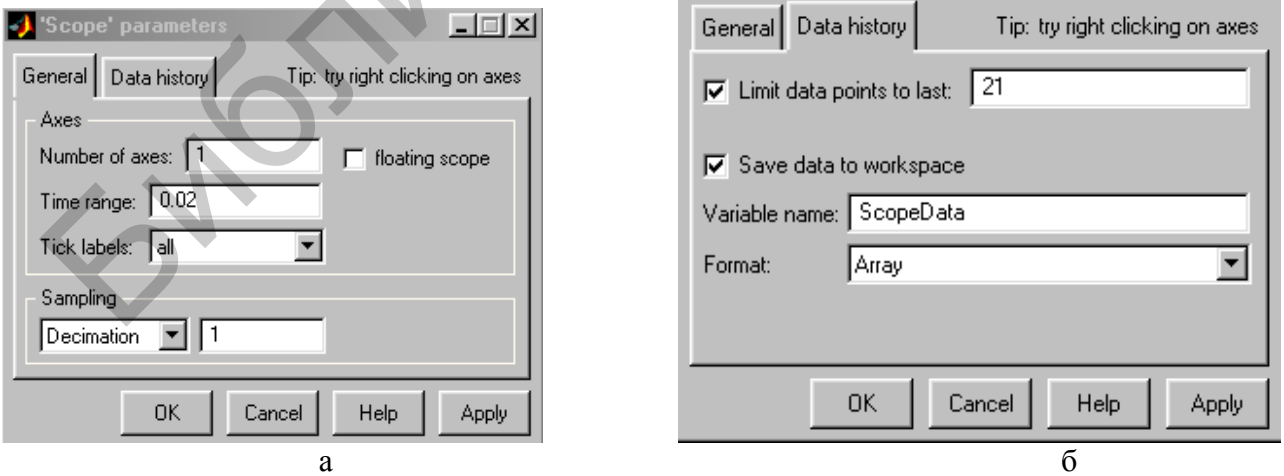


Рис. 20. Окно настройки параметров моделирования

Кроме того, настроим параметры осциллоскопа, щелкнув по кнопке **Parameters** на окне **Scope** (рис. 21, а, б).



а б

Рис. 21. Окна настройки окна Scope

После запуска модели на экране осциллоскопа появится изображение (рис. 22).

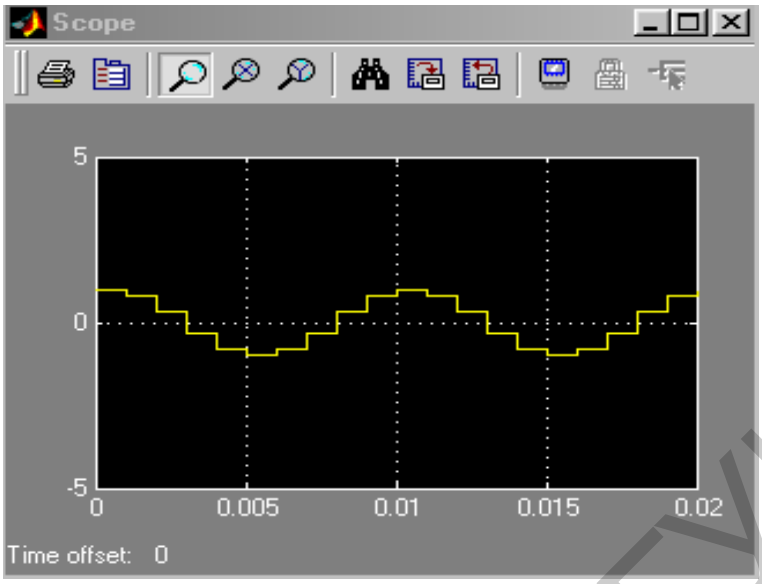


Рис. 22. Результат правильной установки параметров моделирования

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. 23).

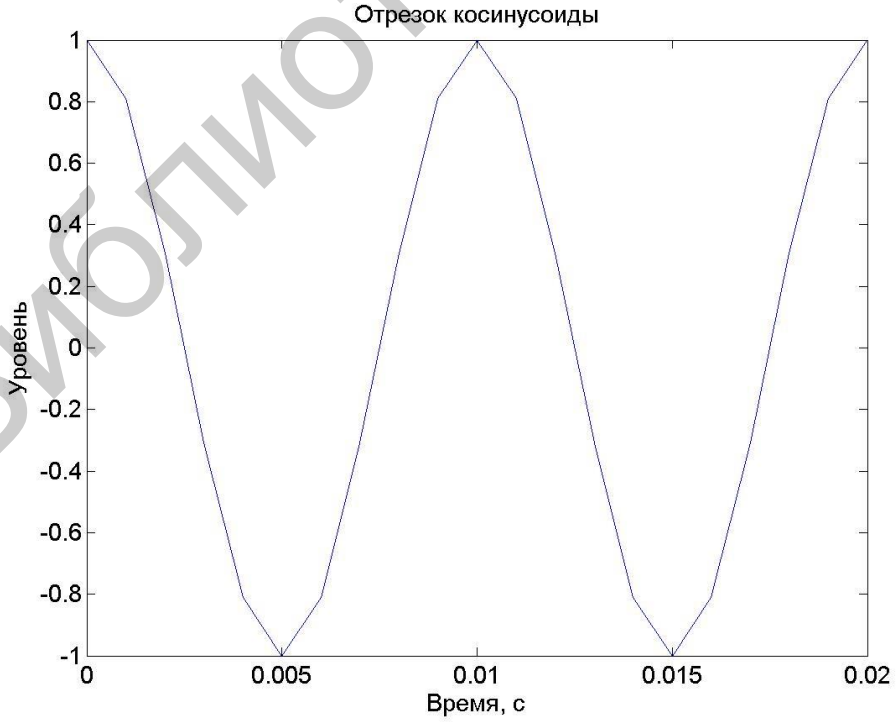


Рис. 23. Результат моделирования сигнала в среде SIMULINK

Сравнивая рис. 23 и рис. 3, 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 F_s 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.

3. Заключительные замечания и задание на выполнение лабораторной работы

Дискретизация аналоговых сигналов – первый шаг на пути решения задачи сопряжения аналоговых устройств и систем с дискретными сигналами.

Моделирование дискретных сигналов можно производить либо в среде MATLAB, либо в среде SIMULINK. Возможно совместное использование этих сред, что увеличивает гибкость инструментария.

Моделирование сигналов в SIMULINK удобно благодаря своей наглядности, однако требует известных навыков задания параметров блоков, из которых конструируется модель.

Важной особенностью моделирования в SIMULINK является очевидное различие понятий «реальное время» и «модельное время». Реальное время – это время, необходимое для проведения вычислений (моделирования). Модельное время – длительность моделируемого процесса.

Модельное время может быть непрерывным (**time-based**) и дискретным (**sample-based**). Непрерывное время рекомендуется использовать при моделировании непрерывных (аналоговых) систем, дискретное – дискретных (цифровых). Контроль результатов моделирования в SIMULINK можно осуществлять как путем построения графиков, так и путем распечатки значений числовых массивов.

Построение графиков удобнее производить в среде SIMULINK. Анализ числовых данных удобнее производить, экспортируя их в рабочее пространство MATLAB.

Задание. Разработать различные варианты решения задачи о прохождении

- одиночного прямоугольного видеоимпульса,
- периодической последовательности этих импульсов

через дифференцирующую и интегрирующую линейные цепи. Для этого построить график входного сигнала, вычислить его спектр и комплексный коэффициент передачи соответствующей линейной цепи, вычислить произведение спектра входного сигнала на комплексный коэффициент передачи, т. е. спектр выходного сигнала, и по нему, вычислив обратное преобразование Фурье, рассчитать форму выходного сигнала. Задачу можно решать как в среде MATLAB, так и в среде SIMULINK.

Для решения поставленной задачи ниже приведена программа, написанная на языке, который понятен пакету MATLAB. Наберите эту программу в ко-

мандном окне и запустите на исполнение. В результате работы программы получим результат, который соответствует анализу прохождения одиночного прямоугольного импульса через интегрирующую цепь. Выполните эту программу несколько раз, изменяя параметры интегрирующей цепи и длительность импульса.

```
> max=500;
>> i=0:max;
>> x=horzcat(zeros(1,50),ones(1,100),zeros(1,max-150))*3;
>>
>> R=5;
>> C=.1;
>>
>> c=fft(x);
>> k=0:250;
>>
>> ck=c(k+1);
>> mk=abs(ck);
>> dk=angle(ck);
>> Kk=1./(1+j*k*R*C);

>> ck=Kk.*ck;
>> zk=abs(ck);
>> fzk=angle(ck);
>>
>> subplot(3,2,1);

>> plot(k(1:20),mk(1:20));
>> subplot(3,2,2);
>> plot(k(1:20),dk(1:20));
>> subplot(3,2,3);

>> plot(k(1:20),zk(1:20));
>> subplot(3,2,4);
>> plot(k(1:20),fzk(1:20));
>> y=ifft(ck);
>> subplot(3,2,5);
>> plot(k,y);
```

После выполнения этой программы замените интегрирующую цепь на дифференцирующую и выполните и запустите модифицированную программу на выполнение. Измените параметры дифференцирующей цепи и снова запустите программу на выполнение. Объясните полученные результаты.

Затем получите у преподавателя задание для анализа прохождения сигнала через линейную цепь с заданными характеристиками цепи и сигнала.

4. Контрольные вопросы

1. Что представляет собой операционная среда системы MATLAB/SIMULINK?
2. Какие опции содержит командное окно системы MATLAB?
3. Какие типы m-файлов вам известны?
4. Из каких компонентов состоит m-файл, оформленный в виде функции?
5. Какие типы данных использует система MATLAB?
6. Верно ли утверждение, что m-файлы представляют собой обычные текстовые файлы?
7. Чем отличаются m-сценарии от m-функций?
8. Назовите основные элементы структуры m-функции.
9. Что представляет собой процесс, называемый дискретизацией?
10. Назовите основные способы, с помощью которых можно генерировать сигналы в системе MATLAB.
11. Чем различаются понятия «время» и «модельное время» в системе MATLAB?
12. Изобразите схему экспортирования данных из среды MATLAB в среду SIMULINK.
13. Приведите структурные схемы генерирования гармонического сигнала в системе MATLAB/SIMULINK.
14. Приведите структурные схемы генерирования сигнала в виде импульса прямоугольной формы в системе MATLAB/SIMULINK.
15. Приведите структурные схемы генерирования сигнала в виде импульса треугольной формы в системе MATLAB/SIMULINK.
16. Приведите структурные схемы генерирования сигнала в виде импульса пилообразной формы в системе MATLAB/SIMULINK.
17. В чем состоят различия анализа прохождения одиночного импульса и периодической их последовательности через интегрирующую или дифференцирующую цепь в системе MATLAB/SIMULINK?

Лабораторная работа №2
ПРИНЦИПЫ МОДУЛЯЦИИ, ДЕМОДУЛЯЦИИ И ФИЛЬТРАЦИИ

Цель работы. Изучить принципы модуляции, демодуляции и фильтрации с использованием пакета MATLAB. В результате выполнения этой работы студенты должны приобрести навыки применения схем беспроводной передачи информации в системах с частотным разделением каналов и написать коды системы MATLAB, нужные для приема речевых сигналов, которые модулируют несущую частоту в различных диапазонах частот. Также нужно уяснить суть частотной отсечки (frequency offset) при демодуляции сигналов.

1. Краткие теоретические сведения

Модуляция, демодуляция и фильтрация составляют суть частотного разделения каналов (frequency division multiplexing, FDM) в системах связи, которое нужно реализовать, выполняя эту работу. Студентам предлагается изучить механизм обработки сигналов при реализации FDM на примерах использования простейших сигналов синусоидальной формы.

1.1. Процесс модуляции

Процесс выполнения приведенных ниже упражнений позволяет освоить механизмы модуляции и демодуляции и понять эффекты, сопровождающие обработку сигналов во временной и частотной областях. Чтобы справиться с поставленной задачей, будем изучать модуляцию и демодуляцию, используя сначала модулирующий сигнал *x* с частотой 1 Гц и модулируемый сигнал *y* с частотой 10 Гц, а затем перейдем на более высокие частоты радиодиапазона, выбирая частоту сигнала *x* равной 10^3 Гц или 10^4 Гц и частоту сигнала *y* равной 100 кГц или 1 МГц и другие частоты по усмотрению каждого студента, выполняющего данную работу.

Рассмотрим сигнал x(t) = cos2πt. Этот сигнал представляет собой синусоидальный сигнал с единичной амплитудой и частотой 1 Гц. Создадим матлабовский файл под названием «modulation.m» и введем следующие команды, чтобы создать и визуализировать *x(t)* во временной области. Результаты работы написанных кодов показаны на рис. 1.

```
<<Fs=1000; %частота дискретизации, измеряемая в Гц
<<t=0:1/Fs:5; %временной вектор, формируемый на промежутке от 0 до 5 с, значения которого отстоят друг от
                %друга на расстоянии 1 мкс, т.е. 0,  0,001, 0.002, 0.003 ….
<<x=cos(2*pi*1*t); % функция косинуса с единичной амплитудой и частотой 1 Гц
<<figure(1) % создаем  рис. 1
<<plot(t,x) % изображаем сигнал x(t) на рис. 1
<<title('временная область сигнала x(t)'); % добавляем название рисунка, которое будет
                                % помещено системой MATLAB вверху над рисунком
<<xlabel('время, с'); % добавляем обозначение «время, с» для x-оси на рис. 1
<<ylabel('амплитуда'); % добавляем обозначение  «амплитуда» для y-оси на рис. 1
<<axis(0 1 -2 2)  % устанавливаем на x-оси диапазон [9, 1] и на y-оси диапазон [-2 2]
```

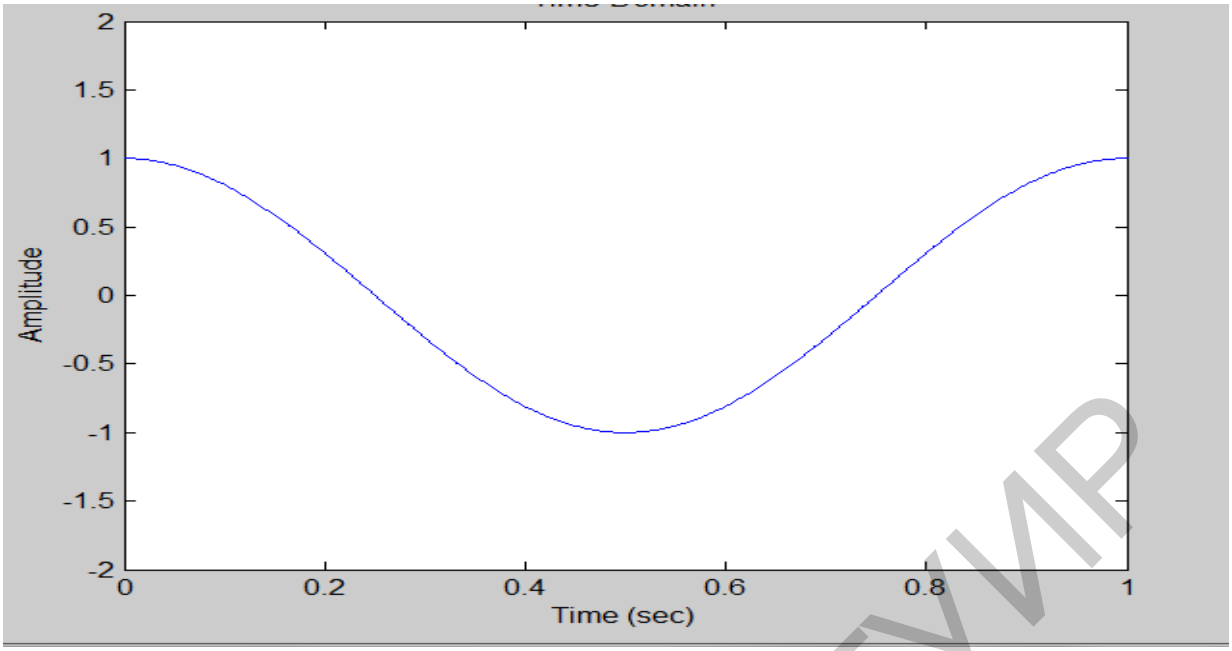


Рис. 1. Форма синусоидального сигнала с единичной амплитудой и частотой 1Гц

Теперь визуализируем амплитуды спектра сигнала *x(t).* Так как сигнал *x(t)* действительный, т. е. описывается функцией действительного переменного, то амплитуда его спектра является четной функцией, т.е. симметрична относительно точки 0 на действительной оси. Поэтому ожидаем увидеть два пика: один на частоте 1 Гц, а другой – на частоте – 1 Гц, соответствующих амплитудам спектральных составляющих спектра *x(t)*. Результаты работы этих команд показаны на рис. 2.

```
<<X=fft(x); % прямое преобразование Фурье-сигнала x(t)
<<N=length(x); % определяем длину частотного вектора x(t)
<<f=[-N/2:N/2-1]*(Fs/N); % создаем частотный вектор f, что простирается
                         % на оси частот от -500 Гц до 500 Гц
<<figure(2) % создаем рис. 2
<<plot(f, abs(fftshift(X(1:N)))); % визуализируем положительную
                                  % и отрицательную части спектра сигнала x(t)
<<axis([-5 5 0 3000]); % устанавливаем диапазон значений на x-оси,
                       %равный [-5 5]  и на y-оси – равный [o 3000]
<<title('частотная область'); % добавляем название рисунка «частотная область», которое
                              % будет помещено системой MATLAB вверху рис. 2
<<xlabel('частота'); % добавляем название x-оси системы координат рис. 2
<<ylabel('амплитуда'); % добавляем название y-оси системы координат рис. 2
```

Теперь промодулируем сигналом *x(t)* сигнал с частотой 10 Гц. Чтобы это сделать, нужно сгенерировать модулируемый сигнал *y(t)=cos(2π*10*t)* и затем создать модулированный сигнал *m(t)=x(t)y(t).* Это можно сделать, выполнив свертку сигналов *x(t)* и *y(t)* в частотной области, но можно использовать и простое тригонометрическое тождество

$$\cos 2\pi f_1 t * \cos 2\pi f_2 t = \frac{\cos[2\pi(f_1 + f_2)t] + \cos[2\pi(f_1 - f_2)t]}{2},$$

записав

$$m(t) = \cos(2 * pi * 1 * t) * \cos(2 * pi * 10 * t) = 0{,}5 * \cos(2 * pi * 11 * t) + 0{,}5 * \cos(2 * pi * 9 * t).$$

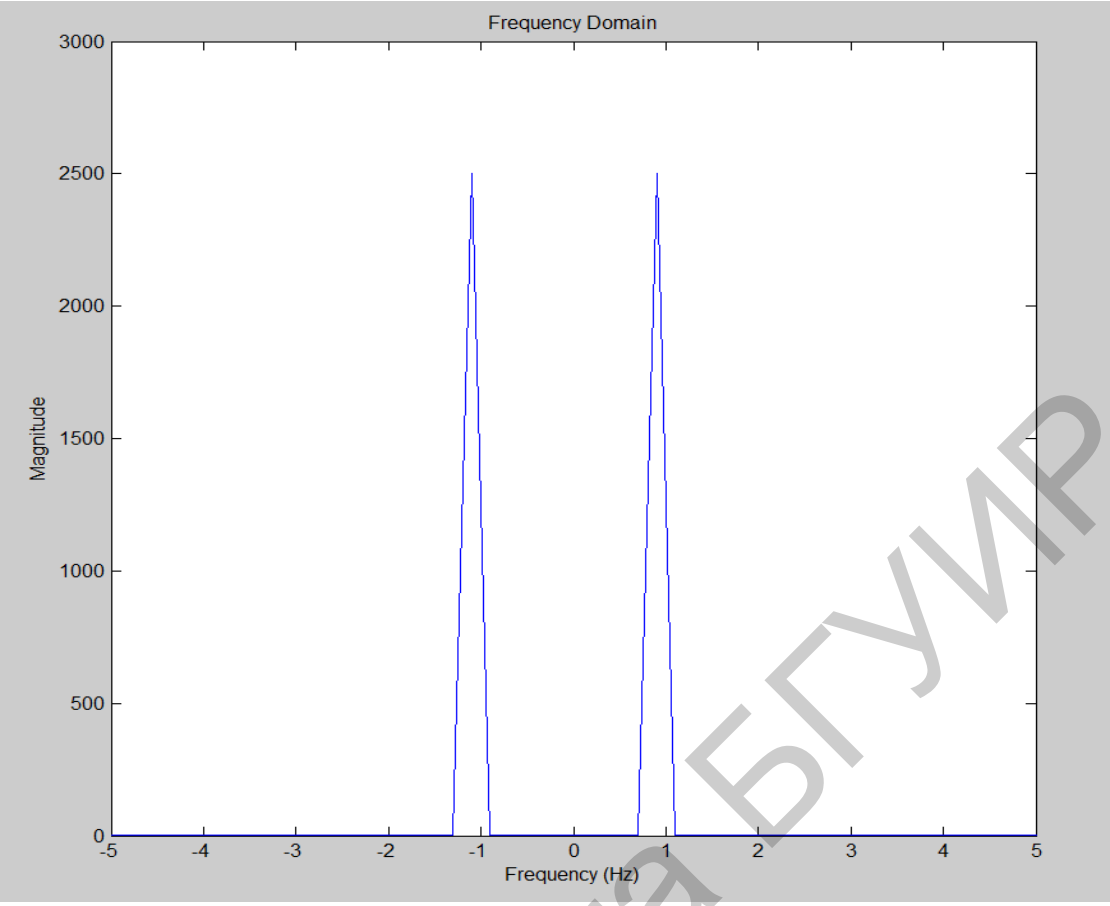


Рис. 2. Спектральные составляющие амплитудно-модулированного сигнала с подавленной несущей частотой

Добавив следующие команды к нашему матлабовскому файлу и пересчитав их, визуализируем во временной области представление модулируемого сигнала *m(t)*. Результаты работы этих команд представлены на рис. 3. Имеем

```
<<y=cos(2*pi*10*t); %создаем модулируемый сигнал, представляющий собой косинусоиду
                    % с единичной амплитудой и частотой 10 Гц
<<m=x.*y; % перемножая модулирующий сигнал x(t) с модулируемым сигналом y(t)
          %создаем модулированный сигнал m(t)
<<figure(3) % создаем рис. 3
<<plot(t,m) % создаем изображение сигнала m(t) на рис. 3
<<title('временная область'); % добавляем название рисунка «временная область», которое
                    % будет помещено системой MATLAB вверху рис. 3
<<xlabel('частота'); % добавляем название x-оси системы координат рис. 3
<<ylabel('амплитуда'); % добавляем название y-оси системы координат рис. 3
<<axis([0 1 -2 2]);  % устанавливаем на x-оси диапазон [0, 1] и на y-оси – диапазон [-2 2]
```

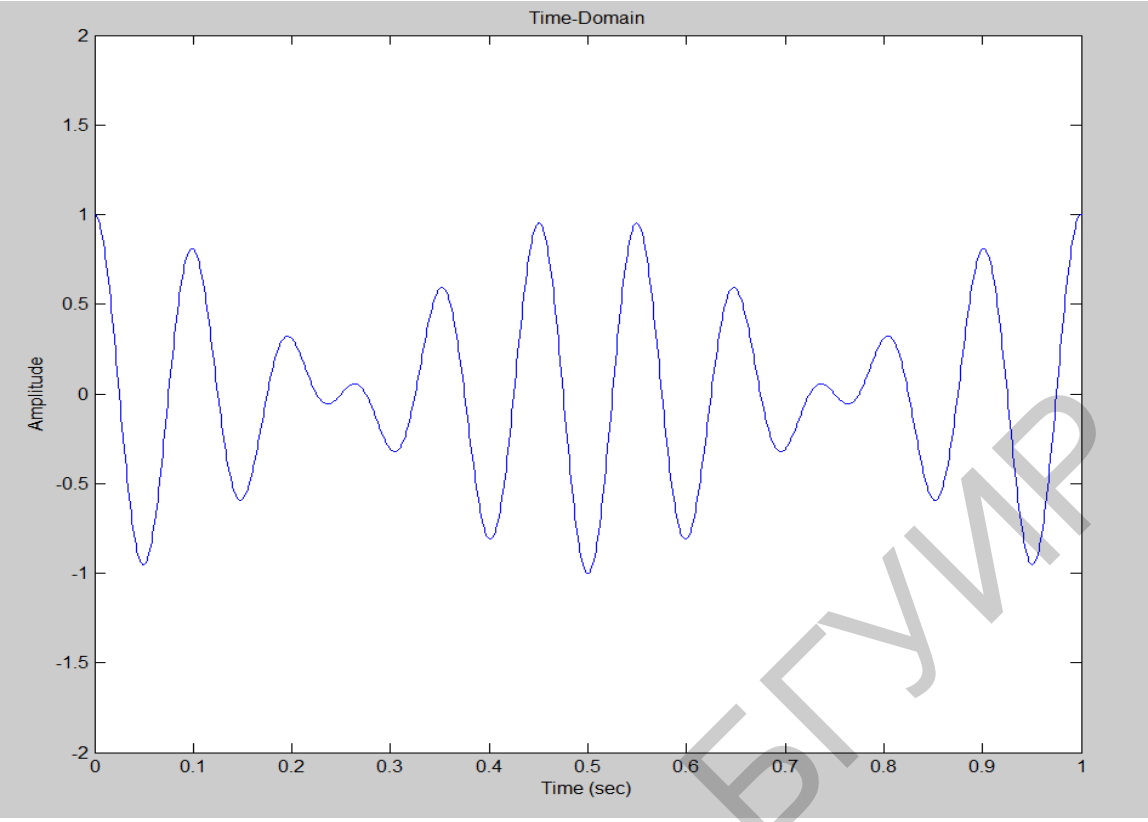


Рис. 3. Форма модулированного по амплитуде сигнала с несущей частотой 10 Гц модулирующим сигналом с частотой 1 Гц

На рис. 4 показаны формы модулирующего сигнала с частотой 1 Гц (кривая изображена пунктирной линией) и модулированного сигнала с несущей частотой 10 Гц (на рис. 4 показана сплошной линией).

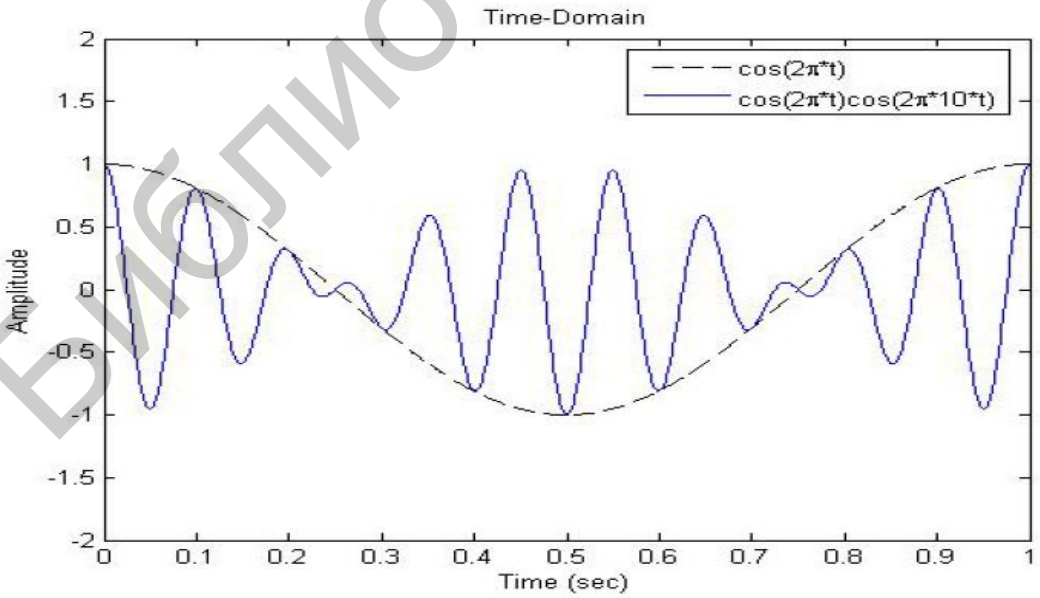


Рис. 4. Сравнение формы колебаний модулирующего сигнала с частотой 1 Гц с модулированным колебанием с несущей частотой 10 Гц

Теперь визуализируем амплитуды спектральных составляющих сигнала *m(t)*. Так как сигнал *m(t)* – действительный, т. е. описывается функцией действительного переменного, то амплитуда его спектра является четной функцией, т. е. симметрична относительно точки 0 на действительной оси. Поэтому ожидаем увидеть две пары пиков: одну на частотах ±9 Гц, а другую – на частотах ±11 Гц, соответствующих амплитудам спектральных составляющих спектра *m(t)*. Результаты работы этих команд показаны на рис. 5, а пояснение сути выполненных процессов иллюстрируется рис. 6.

.

```
<<M=fft(m); % прямое преобразование Фурье-сигнала m(t)
<<f=[-N/2:N/2-1]*(Fs/N); % создаем частотный вектор f, что простирается
                         % на оси частот от -500 Гц до 500 Гц
<<figure(4) % создаем рис. 4
<<plot(f, abs(fftshift(M(1:N)))); % визуализируем положительную
                                  % и отрицательную части спектра сигнала x(t)
<<axis([-20 20 0 1500]); % устанавливаем диапазон значений на x-оси,
                         %равный [-20 20]  и на y-оси – равный [о 1500]
<<title('частотная область'); % добавляем название рисунка «частотная область», которое
                              % будет помещено системой MATLAB вверху рис. 4
<<xlabel('частота'); % добавляем название x-оси системы координат рис. 4
<<ylabel('амплитуда'); % добавляем название y-оси системы координат рис. 4
```

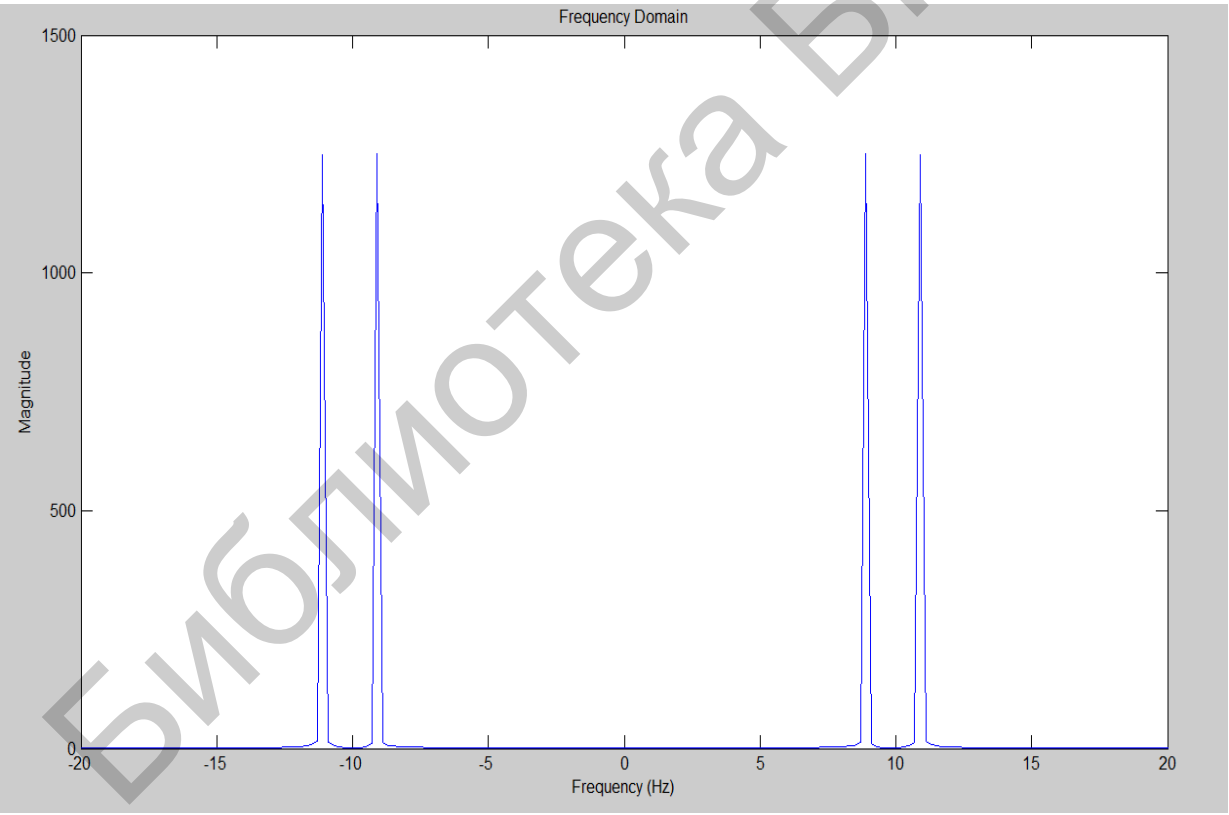


Рис. 5. Спектральная диаграмма модулированного по амплитуде сигналом с частотой 1 Гц сигнала с несущей частотой 10 Гц с подавленной несущей частотой

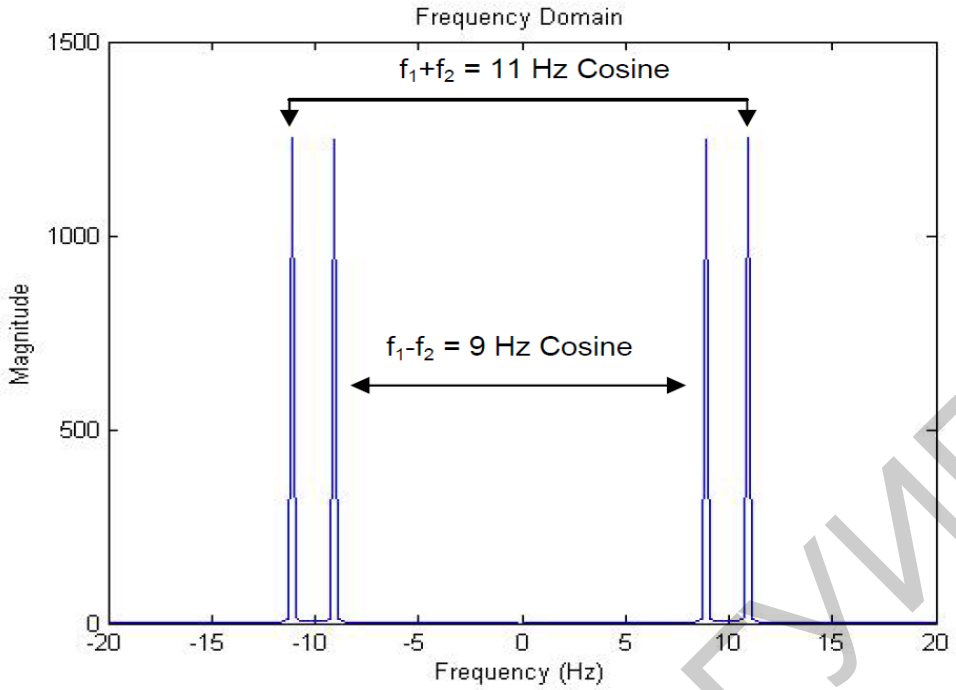


Рис. 6. Пояснение сути полученной спектральной диаграммы, показанной на рис. 5

Таким образом, мы завершили выполнение модуляции сигнала *y(t)* с частотой 10 Гц сигналом *x(t)* с частотой 1 Гц. Этот процесс можно проиллюстрировать диаграммой, показанной на рис .7.

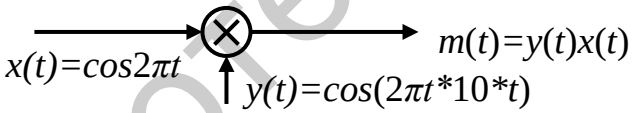


Рис. 7. Структурная схема модулятора

Реализованный процесс называется модуляцией с подавлением несущей частоты. Выполнен этот процесс при 100-процентной модуляции. Для более глубокого понимания сути происходящих при модуляции процессов рекомендуется выполнить его при других значениях коэффициента модуляции, как меньше единицы, так и больше. Проделайте эти операции самостоятельно.

1.2. Демодуляция

Рассмотрим теперь выполнение процесса демодуляции сигнала m(t), чтобы получить содержащийся в нем модулирующий сигнал x(t). Для выполнения этого нужно перемножить сигнал m(t) с сигналом y(t) несущей частоты. Из тригонометрии известно, что результирующий сигнал c(t) примет форму, описываемую выражением

$$c(t) = \frac{1}{4}\cos[2\pi(f_1 - 2f_2)t] + \frac{1}{2}\cos 2\pi f_1 t + \frac{1}{4}\cos[2\pi(f_1 + 2f_2)t]$$

или

$$c(t) = \frac{1}{4}\cos(2\pi * 19*)t) + \frac{1}{2}\cos 2\pi * t + \frac{1}{4}\cos(2\pi * 21 * t).$$

Добавим следующие команды к нашему матлабовскому файлу и пересчитаем их, чтобы визуализировать во временной области представление демодулированного сигнала c(t). Результат выполнения этих команд показан на рис. 8.

```
<<c=m.*y; % перемножение модулированного сигнала m(t) с сигналом y(t)
                % несущего колебания, чтобы получить демодулированный сигнал c(t)
<<figure(5) % создаем рис. 8
<<plot(t,c)  % изображаем сигнал c(t) на рис. 8
<<title('временная область'); % добавляем название рисунка «временная область», которое
                % будет помещено системой MATLAB вверху над рис. 8
<<xlabel('частота'); % добавляем название x-оси системы координат рис. 8
<<ylabel('амплитуда'); % добавляем название y-оси системы координат рис. 8
<<axis([0 1 -2 2]);  % устанавливаем на x-оси диапазон [0, 1] и на y-оси – диапазон [-2 2]
```

Теперь визуализируем амплитудный спектр сигнала *c(t)*. Так как сигнал *c(t)* – действительный, т. е. описывается функцией действительного переменного, то амплитуда его спектра является четной функцией, т. е. симметрична относительно точки 0 на действительной оси. Поэтому ожидаем увидеть пики, показанные на рис. 9, на частотах ±1 Гц, ±19 Гц и ±21 Гц, соответствующих амплитудам спектральных составляющих спектра *x(t)*. Добавим следующие команды к нашему матлабовскому файлу. Результаты работы этих команд показаны на рис. 10, а пояснение сути выполненных процессов иллюстрирует рис. 11.

```
<<C=fft(c);  % прямое преобразование Фурье-сигнала c(t)
<<f=[-N/2:N/2-1]*(Fs/N); % создаем частотный вектор f, что простирается
                % на оси частот от -500 Гц до 500 Гц
<<figure(4) % создаем рис. 8
<<plot(f, abs(fftshift(M(1:N)))); % визуализируем положительную
                % и отрицательную части спектра сигнала x(t)
<<axis([-20 20 0 1500]); % устанавливаем диапазон значений на x-оси,
                %равный [-20 20]  и на y-оси – равный [o 1500]
<<title('частотная область'); % добавляем название рисунка «частотная область», которое
                % будет помещено системой MATLAB вверху над рис. 10
<<xlabel('частота'); % добавляем название x-оси системы координат рис.10
<<ylabel('амплитуда'); % добавляем название y-оси системы координат рис. 10
```

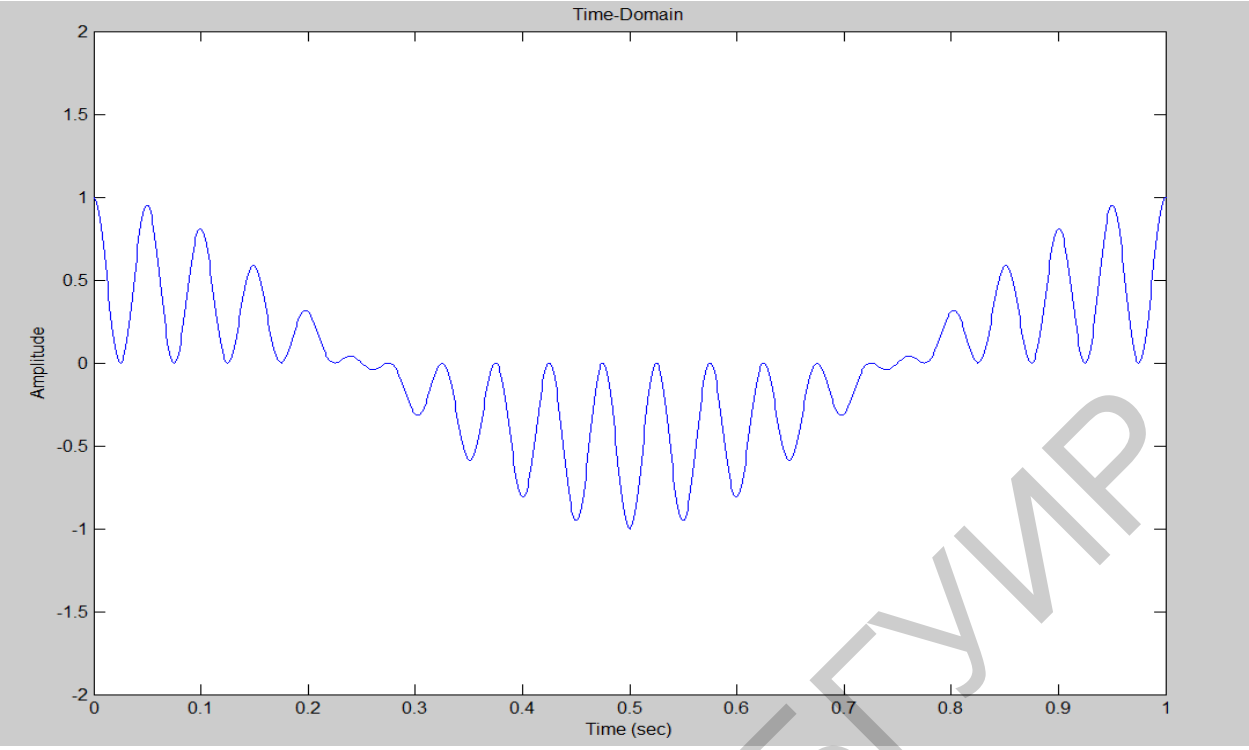


Рис. 8. Представление во временной области демодулированного сигнала

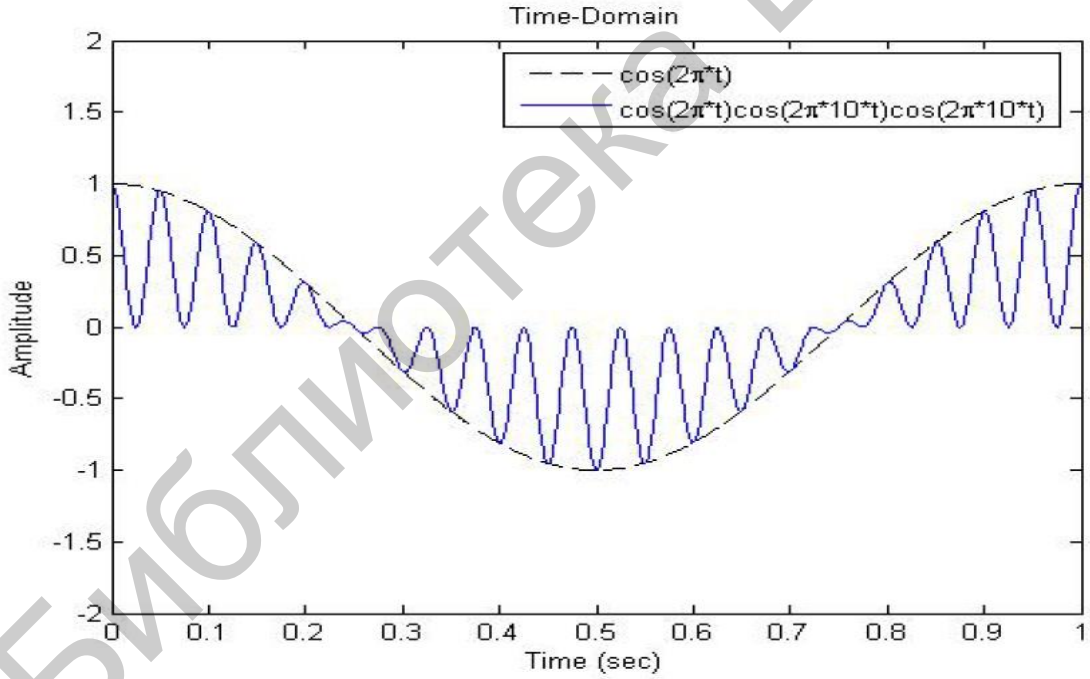


Рис. 9. Пояснение результатов процесса демодуляции, показанного на рис. 9 (пунктирной линией показан модулирующий сигнал с частотой 1 Гц, а сплошная кривая отображает наличие частот 19 и 21 Гц в спектре демодулированного сигнала)

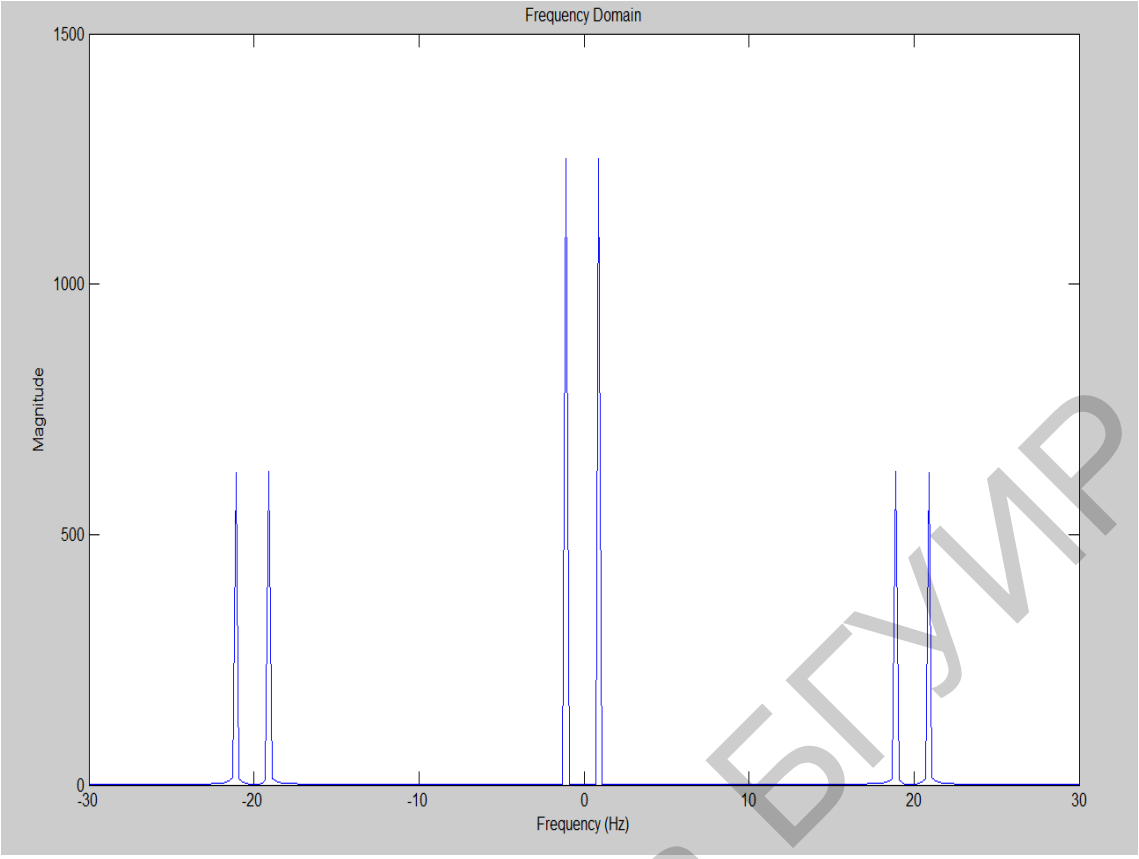


Рис. 10. Спектральное представление демодулированного сигнала

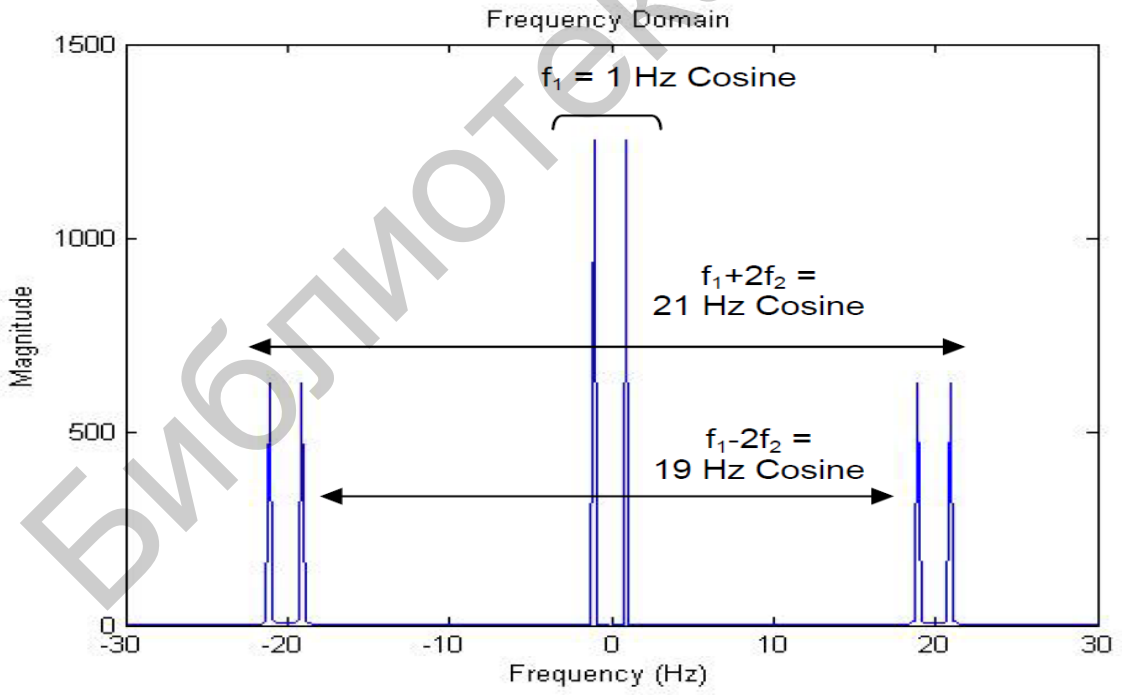


Рис. 11. Пояснение внутренней структуры демодулированного сигнала

Заметим, что мы не восстановили сигнал *x(t)*, так как вместе с сигналом *x(t)* был восстановлен еще ряд сигналов более высоких частот (19 Гц и 21 Гц). Чтобы устранить избыточные высокочастотные компоненты, нужно выполнить процедуру

фильтрации, т. е. создать фильтр, пропускающий только низкие частотные компоненты. Следующий этап работы посвящен выполнению этой фильтрации.

1.3. Фильтрация демодулированного сигнала

Исследуемый в данной работе демодулятор можно описать разностным уравнением

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + ... + a_N y[n-M] + b_0 y]n] + b_1 y[n-1] + ... + b_N y[n-N],$$

или в более компактной форме

$$y[n] = \sum_{j=1}^{M} a_j x[n-j] + \sum_{l=0}^{N} b_l x[n-l].$$

Выходной сигнал описываемого демодулятора определяется как текущим значением сигнала, так и значениями сигналов, поступивших на вход модулятора во время предшествующих тактов его работы. В этой лабораторной работе будем использовать разностное уравнение, которое соответствует FIR-фильтру, т. е. фильтру с конечной импульсной характеристикой, которая описывается разностным уравнением, коэффициенты которого $a_j = 0$, т. е. фильтрацию будем осуществлять фильтром, описываемым уравнением

$$y[n] = \sum_{l=0}^{N} b_l x[n-l].$$

Создадим фильтр низких частот 128 порядка с частотой среза f_c = 25 Гц, чтобы выделить частотную компоненту 10 Гц из сигнала, представленного выражением

$x(n)$=sin(2π*10*n*Ts)+sin(2π*50*n*Ts)+sin(2π*90*n*Ts).

Создадим файл «filtering.m» и поместим в него следующие команды, необходимые для создания сигнала $x(n)$ и представления его во временной области. Результаты работы этих команд показаны на рис. 12.

```
Fs = 1000; % Sample Frequency in Hz
t = 0:1/Fs:5; % Time vector form 0-5 seconds in increments of
% 0.001 seconds (eg. 0 0.001, 0.002, 0.003 ...)
x = sin(2*pi*10*t) + sin(2*pi*50*t)+sin(2*pi*90*t);
figure(1) % Create Figure 1
plot(t,x) % Plot Signal x(n) in Figure 1
title('Time-Domain'); % Add Title Time-Domain to Figure 1
xlabel('Time (sec)'); % Add Time (sec) as x-axis label of Figure 1
ylabel('Amplitude'); % Add Amplitude as y-axis label of Figure 1
axis([0 0.5 -4 4]) % X-axis range [0-0.5] sec and y-axis range [-4 4]
```

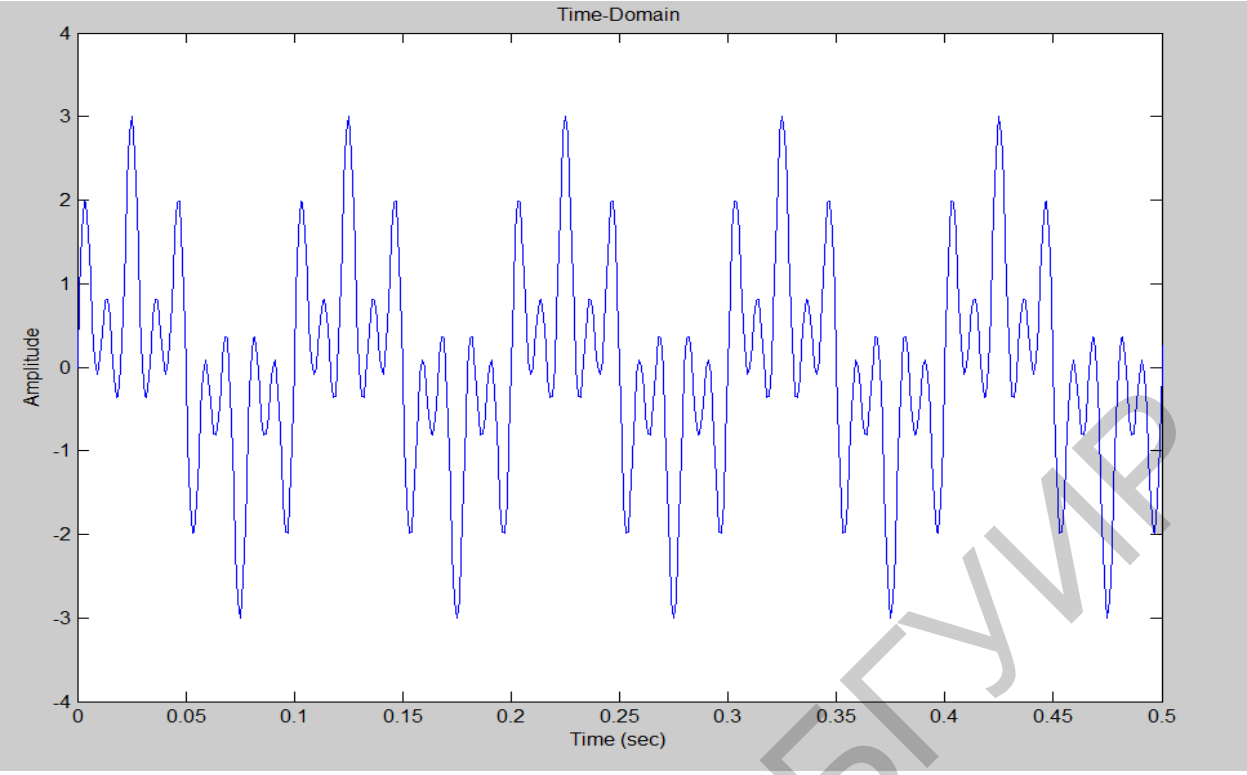


Рис.12. Временная диаграмма анализируемого сигнала

Теперь визуализируем распределение амплитуд спектральных составляющих сигнала *x(n).* Так как сигнал *x(n)* – действительный, то его амплитудный спектр описывается функцией действительного переменного, симметричной относительно точки 0 на оси абсцисс. Ожидаем увидеть пики на частотах ±10 Гц, ±50 Гц и ±90 Гц. Добавим следующие коды к файлу «filtering.m» и выполним его, чтобы визуализировать амплитуды спектра сигнала *x(n).* Результаты выполнения этих команд представлены на рис. 13. Представим более наглядно полученный спектр сигнала на рис. 14, т. е. представим сигнал *x(n)* в частотной области.

```
X = fft(x); % perform FFT on signal x(t)
N = length(x); % Make frequency vector f
f = [-N/2:N/2-1]*(Fs/N); % that spans frequencies from -500 Hz to 500 Hz
figure(2) % Create Figure 2
plot(f,abs(fftshift(X(1:N)))) % Plot positive and negative spectrum of x(t)
axis([-5 5 0 3000]); % Set x-axis range [-5 5] Hz and y-axis [0 3000]
title('Frequency Domain'); % Add Frequency Domain as title of Figure 2
xlabel('Frequency (Hz)'); % Add Frequency (Hz) as x-axis label of Figure 2
ylabel('Magnitude'); % Add Magnitude as y-axis label of Figure 2
```

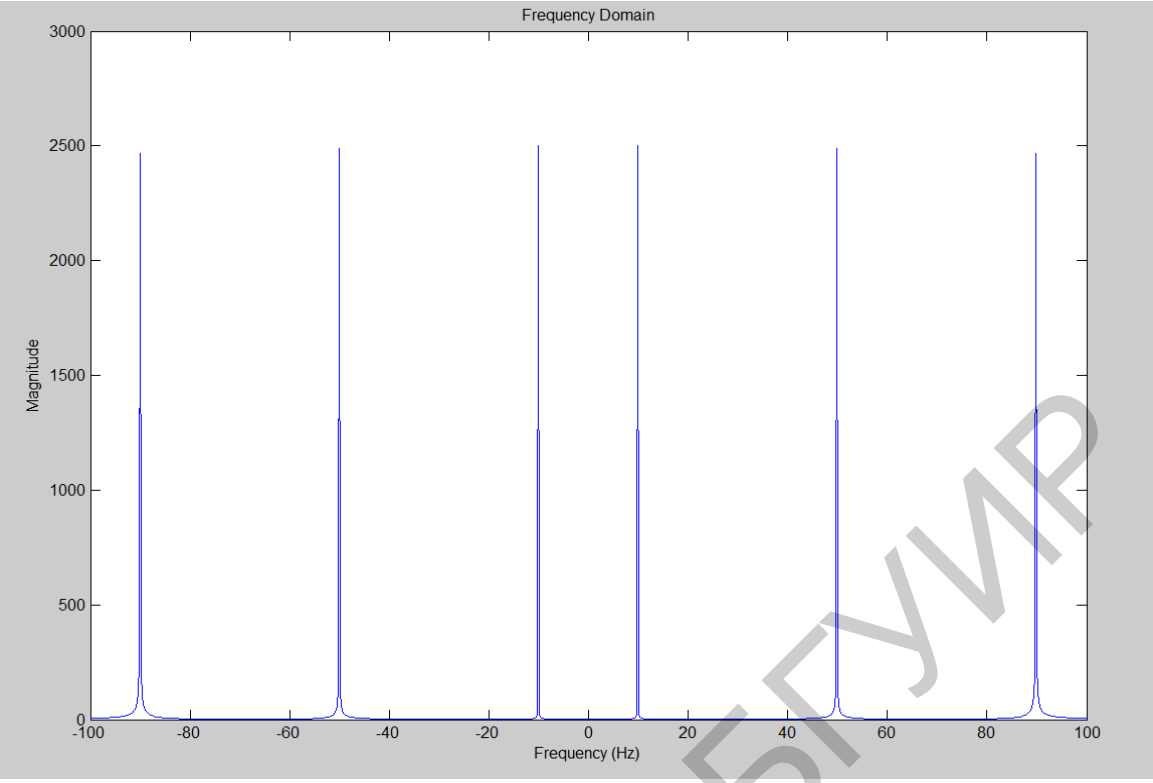


Рис. 13. Спектральные составляющие сигнала $x(n)$

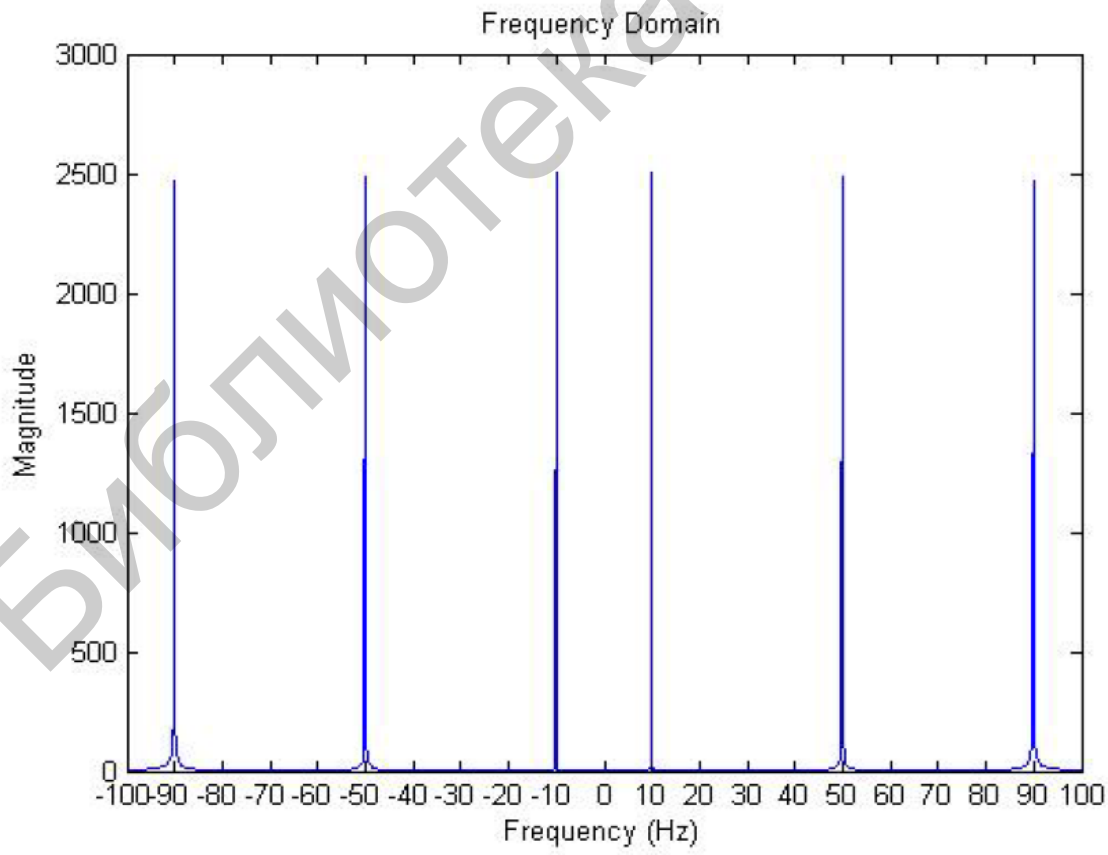


Рис. 14. Наглядное представление спектра сигнала $x(n)$

Теперь создадим ФНЧ, чтобы выделить компоненту с частотой 10 Гц из спектра сигнала x(n). Этот фильтр будет 128 порядка и иметь частоту среза fc = = 25 Гц. Перед выполнением кода напомним некоторые команды, которые будем использовать для создания фильтров и визуализации их частотных откликов.

Для создания фильтра используем команду fir1, имеющую формат

B_coefficients=fir1(Filter_Order, Cuttoff_Frequency, ‘low’).

Переменная Filter_Order отражает порядок используемого фильтра (в данном случае этот порядок равен 128). Переменная Cuttoff_Frequency относится к частоте среза (в данном случае fc= 25 Гц.). Аргумент 'low' служит для указания типа создаваемого фильтра (в данном случае создается ФНЧ). После выполнения этой команды переменная B_coefficients будет представлять собой вектор, элементы которого являются коэффициентами b_l разностного уравнения, соответствующего ФНЧ. Чтобы получить частотный отклик фильтра, используем команду freqz. В общем случае эта команда рассчитывает частотный отклик фильтра, характеризующегося коэффициентами a_j и b_l. В нашем случае мы используем только коэффициенты b_l. Запишем следующие команды, чтобы вызвать функцию freqz:

H=freqz(B_coefficients, [1], ‘whole’);
F=(=256:255)*Fs/512;

Переменная B_coefficients – это вектор b_l коэффициентов, созданных командой fir1. Аргумент [1] относится к команде freqz и служит для расчета частоты отклика для положительных и отрицательных частот, определяемых переменной F. Добавим следующие команды в файл «filtering.m», чтобы создать ФНЧ и визуализировать его частотный отклик. Мы должны получить рисунок, показанный на рис. 15. Приведем также для иллюстрации изображение спектра в логарифмическом масштабе, показанное на рис. 16.

```
Filter_Order = 128; % Set filter order to be 128
Cuttoff_Frequency = 25*(2/Fs); % Set Cuttoff Frequecy to 25 Hz
% Following are commands for creating filter and filter frequency response
B_Coefficients= fir1(Filter_Order, Cuttoff_Frequency,'low');
H = freqz(B_Coefficients, [1], 'whole');
F=(-256:255)*Fs/512;
figure(3) % Create Figure 3
plot(F,abs(fftshift(H))) % Plot positive and negative frequencies
axis([-50 50 0 2]); % X-axis range [-50 50] Hz and y-axis [0 2]
title('Frequency Response'); % Add Frequency Response as title
xlabel('Frequency (Hz)'); % Add Frequency (Hz) as x-axis label
ylabel('Magnitude'); % Add Magnitude as y-axis label
```

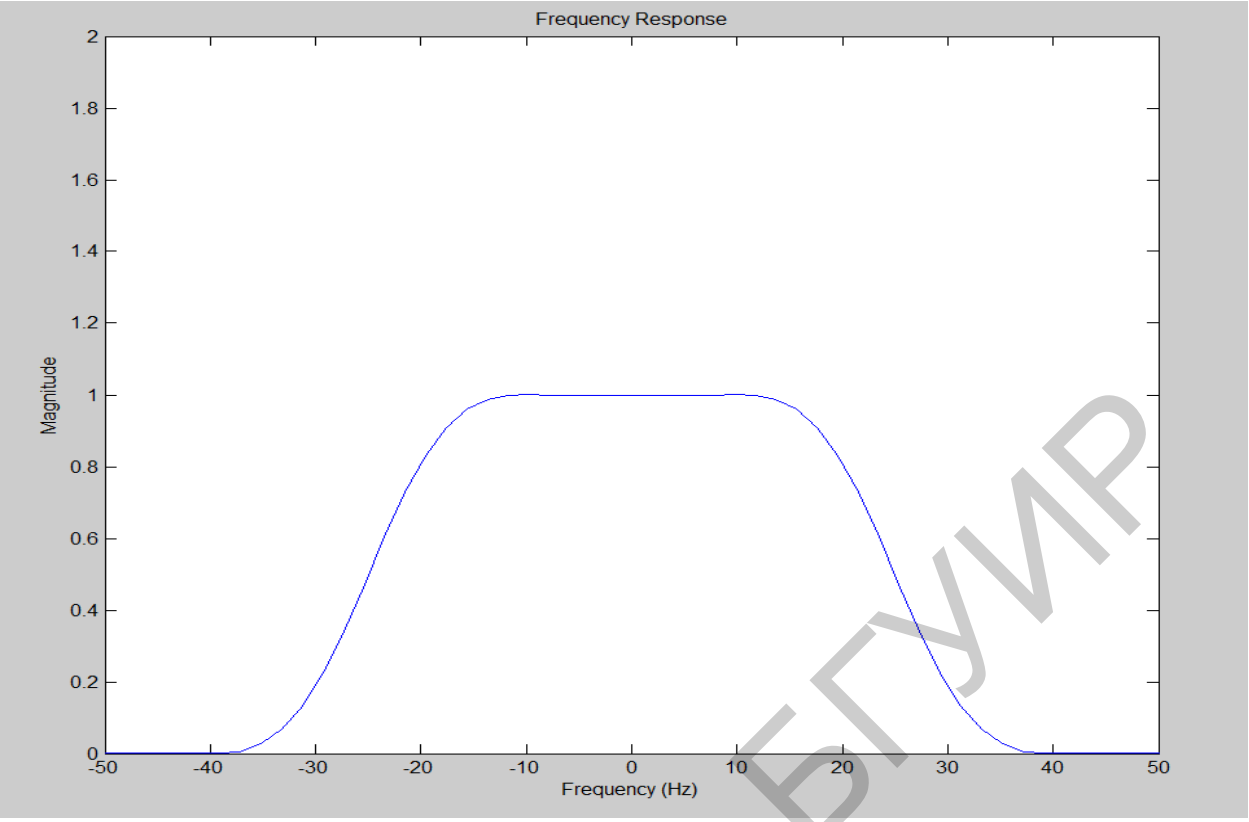


Рис. 15. Частотный отклик ФНЧ

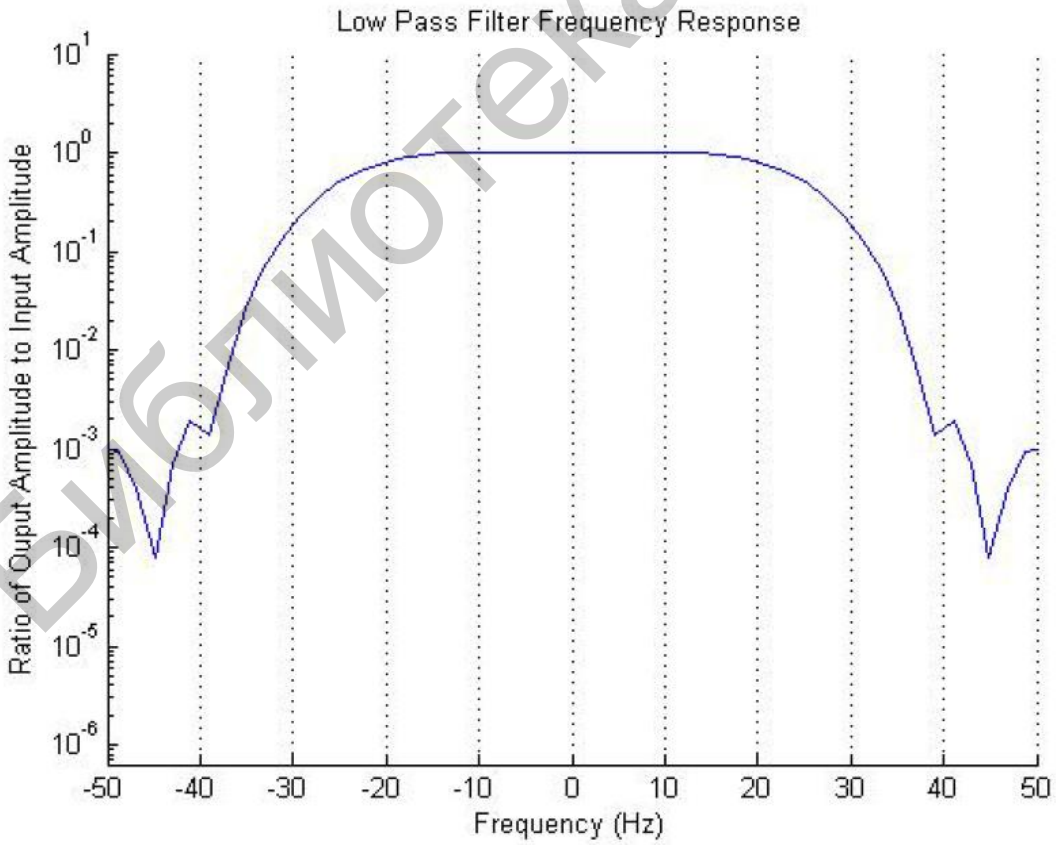


Рис. 16. Частотный отклик ФНЧ в логарифмическом масштабе

Используем ФНЧ для фильтрации сигнала x(n), применив команду filter, которая записывается в формате

Output_signal=filter(B_Coefficients, [1], input_signal);

Переменная B_Coefficients – это вектор b_l коэффициентов разностного уравнения, представляющего ФНЧ, созданный командой fir1. Аргумент [1] относится к команде filter, что не имеет коэффициентов a_i в разностном уравнении. Переменная input_signal ссылается на входной сигнал, который должен быть отфильтрован. После выполнения этой команды переменная output_signal сохранит отфильтрованный сигнал. Добавим следующие коды к нашему файлу, чтобы отфильтровать сигнал x(n). Мы должны получить изображение, подобное показанному на рис. 17.

```
yLPF = filter(B_Coefficients,[1],x); %Apply the filter to the signal x(n)
figure(4) % Create Figure 4
plot((0:length(yLPF)-1)/Fs, yLPF) % Plot Signal y(n) in Figure 4
title('Time-Domain'); % Add Title Time-Domain
xlabel('Time (sec)'); % Add Time (sec) as x-axis label
ylabel('Amplitude'); % Add Amplitude as y-axis label
axis([.1 1 -1 1]) % Set x and y axis range
```

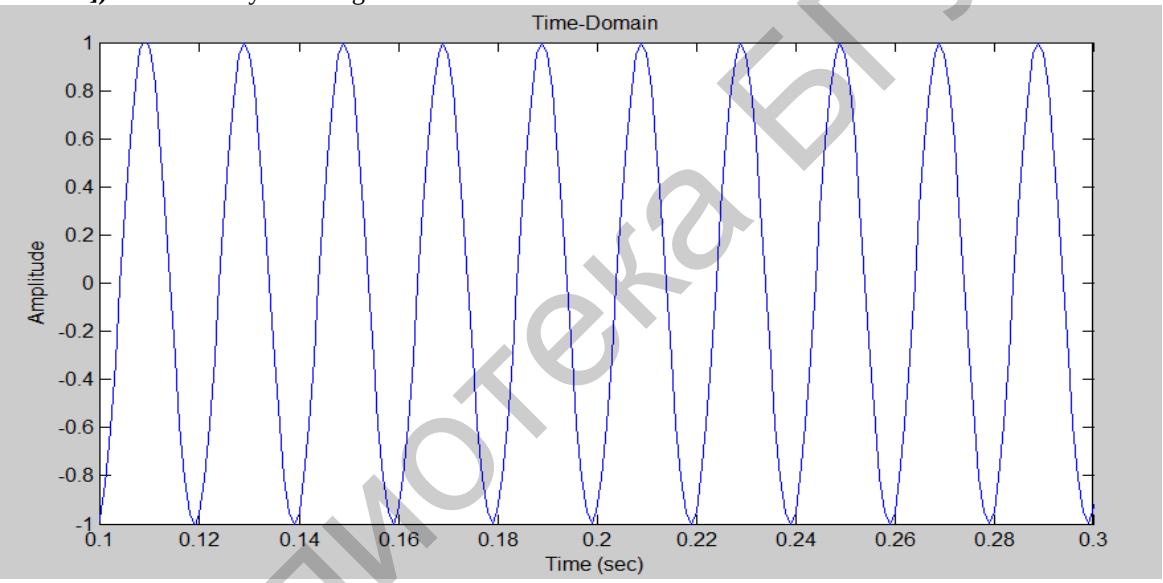


Рис. 17. Отфильтрованный полезный сигнал сообщения

Теперь спроектируем полосовой пропускающий фильтр, чтобы выделить компоненту с частотой 50 Гц из сигнала:

$x(n)$= sin(2π*10*n*Ts)+sin(2π*50*n*Ts)+sin(2π*90*n*Ts).

Полосовой фильтр будет иметь порядок 128 и частоты отсечки $f_{c.low}$=35 Гц и $f_{c.high}$=65 Гц. Чтобы спроектировать полосовой пропускающий фильтр, используем команду fir1, имеющую формат

B_Coefficients=fir1(Filter_Order, [Cuttoff_Frequency_Low
Cuttoff_Frequency_High])

Переменная Filter_Order отражает порядок используемого фильтра (в данном случае этот порядок равен 128). Переменные Cuttoff_Frequency_Low и Cuttoff_Frequency_High относятся к частотам отсечки, в нашем случае равным $f_{c.low}$=35 Гц и $f_{c.high}$=65 Гц. После выполнения этой команды переменная B_Coefficients будет представлять собой вектор, элементами которого будут ко-

эффициенты b_l разностного уравнения, что описывает полосовой фильтр. Добавим следующие коды в файл «filtering.m», чтобы создать полосовой фильтр и визуализировать его частотный отклик. Мы должны увидеть результаты, представленные на рис. 18. Добавим для наглядности рис. 19, который дает представление о спектре в логарифмическом масштабе.

```
Filter_Order = 128; % Set filter order to be 128
Cuttoff_Frequency_Low = 35*(2/Fs); % Set fc,low to 35 Hz
Cuttoff_Frequency_High = 65*(2/Fs); % Set fc,high = 65 Hz
% Following are commands for creating filter and filter frequency response
B_Coefficients= fir1(Filter_Order, [Cuttoff_Frequency_Low Cuttoff_Frequency_High]);
H = freqz(B_Coefficients, [1], 'whole');
F=(-256:255)*Fs/512;
figure(5) % Create Figure 5
plot(F,abs(fftshift(H))) % Plot positive and negative spectrum of filter
axis([-100 100 0 2]); % X-axis range [-100 100] Hz and y-axis [0 2]
title('Frequency Response'); % Add Frequency Response as title
xlabel('Frequency (Hz)'); % Add Frequency (Hz) as x-axis label
ylabel('Magnitude'); % Add Magnitude as y-axis label
```

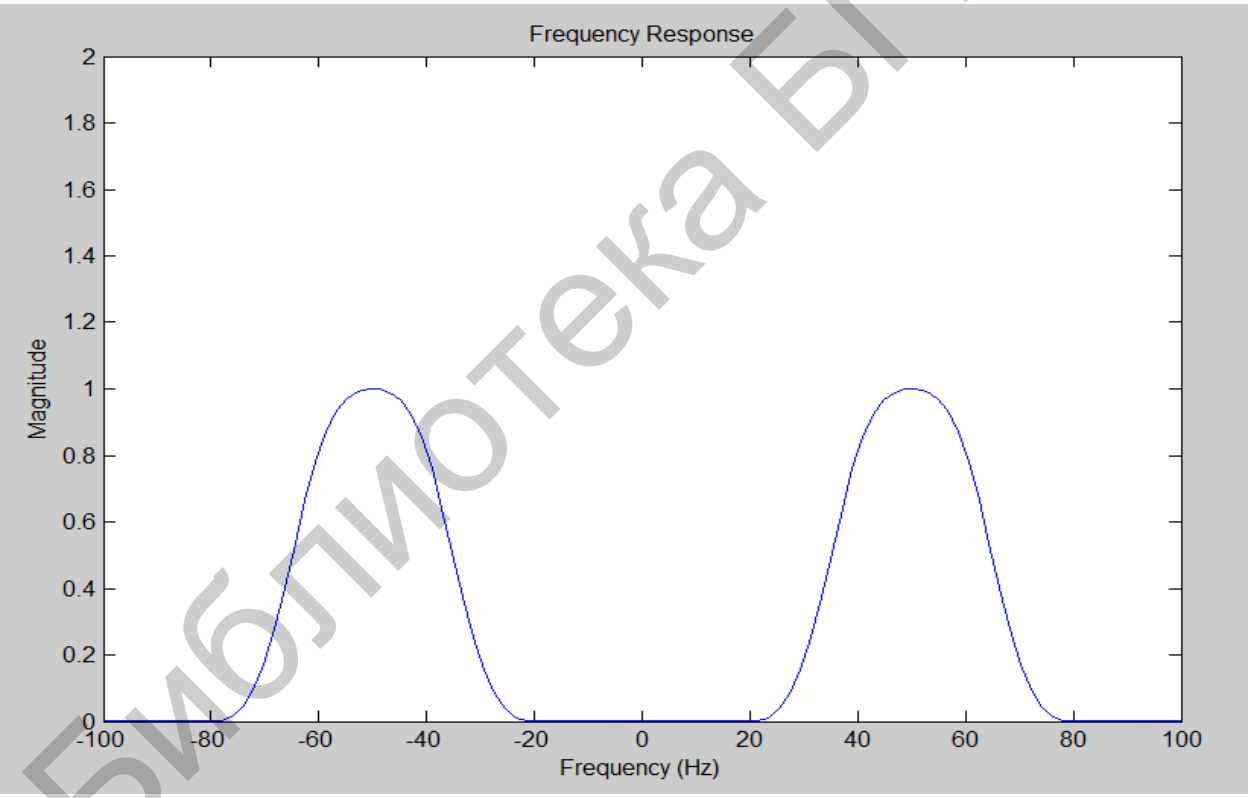


Рис. 18. Частотный отклик полосового пропускающего фильтра

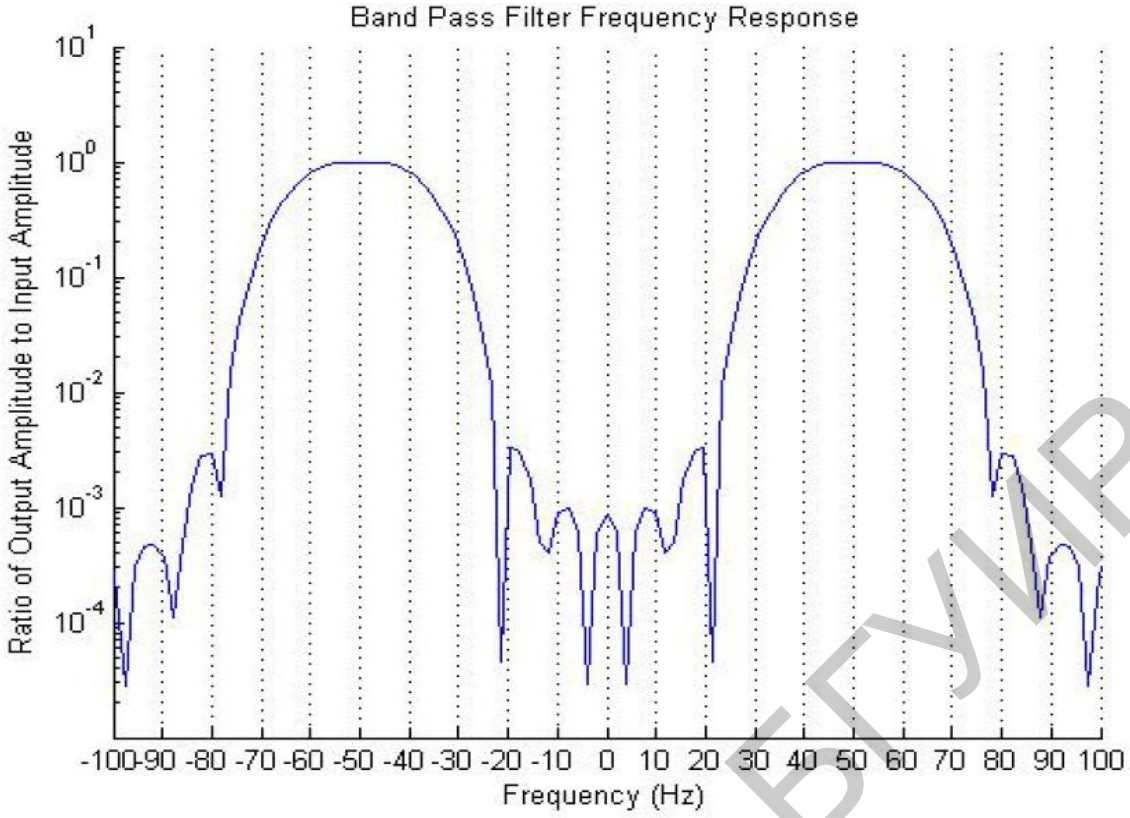


Рис. 19. Частотный отклик полосового фильтра в логарифмическом масштабе

Осталось применить полосовой фильтр к сигналу *x(n)*. Нет никаких изменений в подходе, который мы использовали при создании команды filter. Добавим следующие коды в файл «filtering.m», чтобы отфильтровать сигнал x(n). Мы должны увидеть временную диаграмму, представленную на рис. 20.

```
yHBPF = filter(B_Coefficients,[1],x); %Apply the filter to the signal x(n)
figure(6) % Create Figure 6
plot((0:length(yHBPF)-1)/Fs, yHBPF) % Plot Signal y(n) in Figure 6
title('Time-Domain'); % Add Title Time-Domain to Figure 6
xlabel('Time (sec)'); % Add Time (sec) as x-axis label of Figure 6
ylabel('Amplitude'); % Add Amplitude as y-axis label of Figure 6
axis([.1 .3 -1 1]) % X-axis range [0.1 .3] sec and y-axis range [-1 1]
```

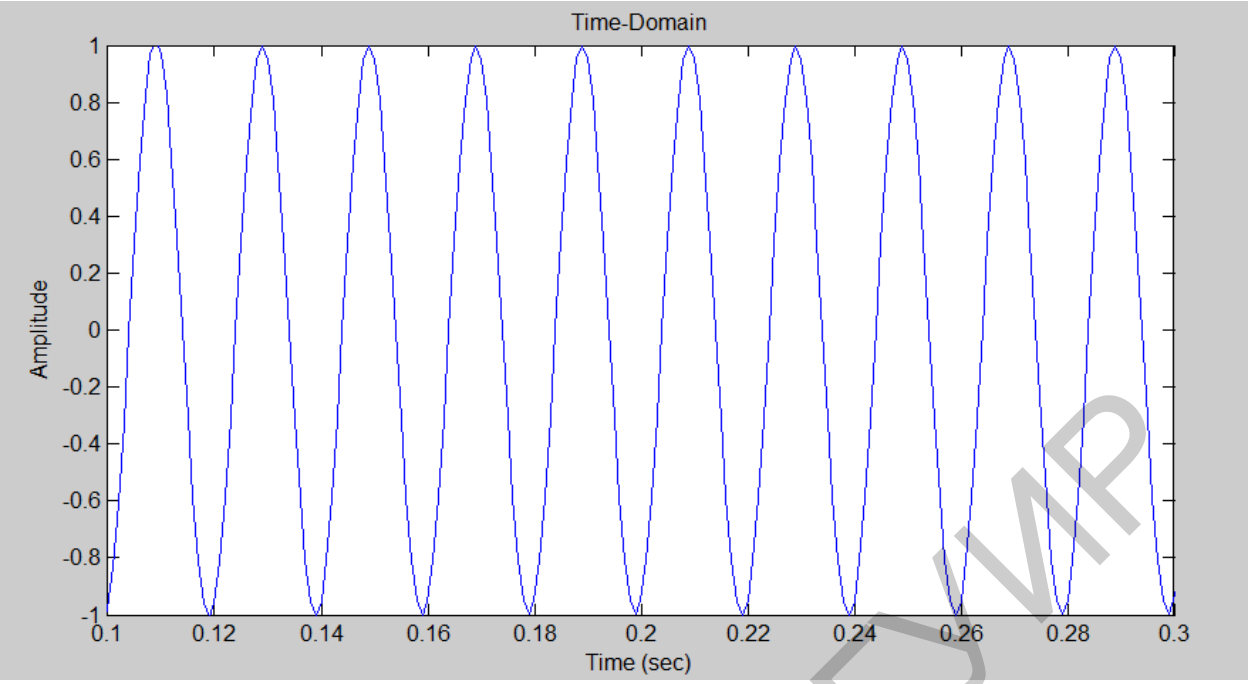


Рис. 20. Выходной сигнал полосового пропускающего фильтра

1.4. Изучение принципа работы технологии частотного разделения каналов

Технологии частотного разделения каналов (Frequency Division Multiplexing, FDM) позволяют передавать несколько информационных сигналов одновременно по одному каналу связи. FDM выделяет каждому информационному сигналу определенную частотную область внутри канала связи. Эти частотные области выбираются так, чтобы они не перекрывались. Рис. 21 иллюстрирует передачу двух информационных сигналов, используя FDM.

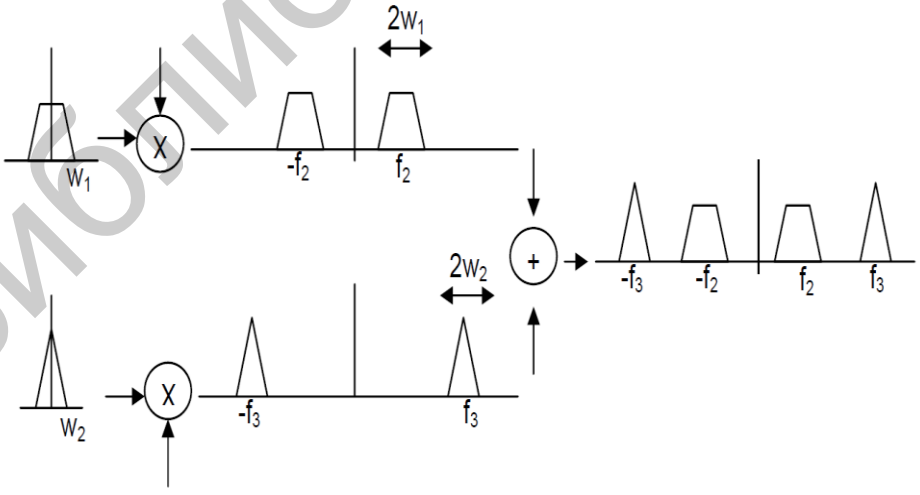


Рис. 21. Принцип частотного разделения каналов

На рис. 22 имеем два информационных сигнала: один с трапецеидальной формой спектра шириной w_1, другой – со спектром треугольной формы шириной w_2. Чтобы передать по каналу связи эти два сигнала, нужно промодулировать ими

несущие f_1 и f_2, находящиеся на достаточно большом расстоянии друг от друга, таком, чтобы спектры полученных в результате радиосигналов не перекрывались. Таким образом, на частоту f_1 переносится трапецеидальный спектр, а частоту f_2 – треугольный спектр. Складывая полученные радиосигналы в сумматоре, получаем суммарный сигнал передачи (transmission signal), который может переносить спектры обоих сигналов одновременно по каналу связи.

На приемном конце канала связи суммарный сигнал передачи подвергается обработке, которая включает устранение нежелательных спектральных компонент из принятого сигнала и демодуляцию, выделяя исходные спектральные компоненты шириной w_1 и w_2. На рис. 22 показано, что нужно сделать, чтобы выделить сигнал с трапецеидальным спектром.

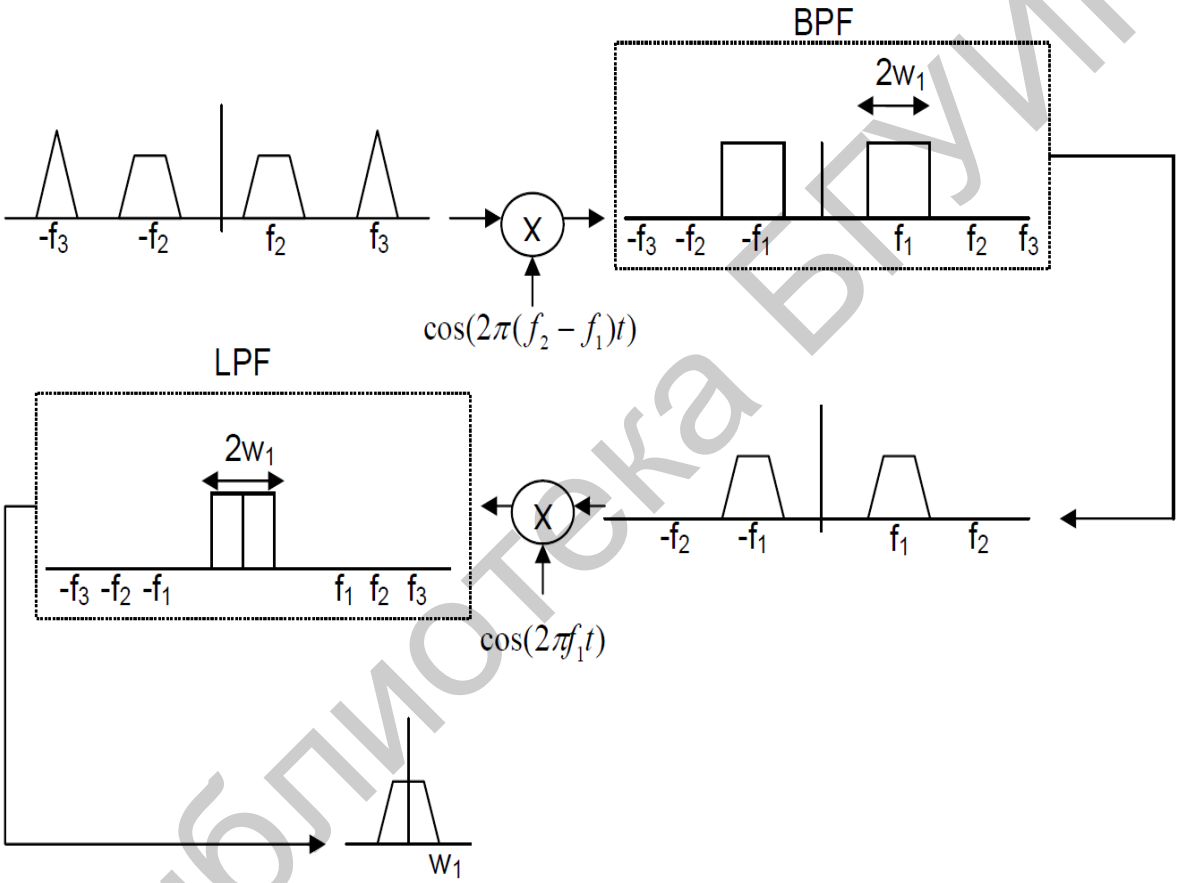


Рис. 22. Процедура выделения сигнала с трапецеидальным спектром

Видим, что сигнал передачи модулируется так, что спектральные компоненты располагаются в окрестности центральной частоты полосового пропускающего фильтра. В этом случае трапецеидальный спектр сдвигается так, чтобы он разместился симметрично на частоте f_1. Полосовой фильтр содержит только спектральные компоненты информационного сигнала (трапецеидальные) и устраняет спектральные компоненты информационного сигнала с треугольной формой спектра.

Далее, трапецеидальный спектр демодулируется к исходному спектру и ФНЧ устраняет все более высокие частоты, которые появляются в результате

демодуляции (это частоты $-2f_1$ и $2f_1$). В результате получается информационный сигнал с трапецеидальным спектром.

Эта же процедура используется для выделения информационного сигнала с треугольным спектром. Шаги, необходимые для выделения сигнала с треугольным спектром, показаны на рис. 23.

В конечном счете ключевая ступень в приеме информационного сигнала и выделение его из сигнала передачи при FDM – это выделение модулированных частот в приемнике, что центрирует спектр информационного сигнала в полосе рабочих частот полосового пропускающего фильтра.

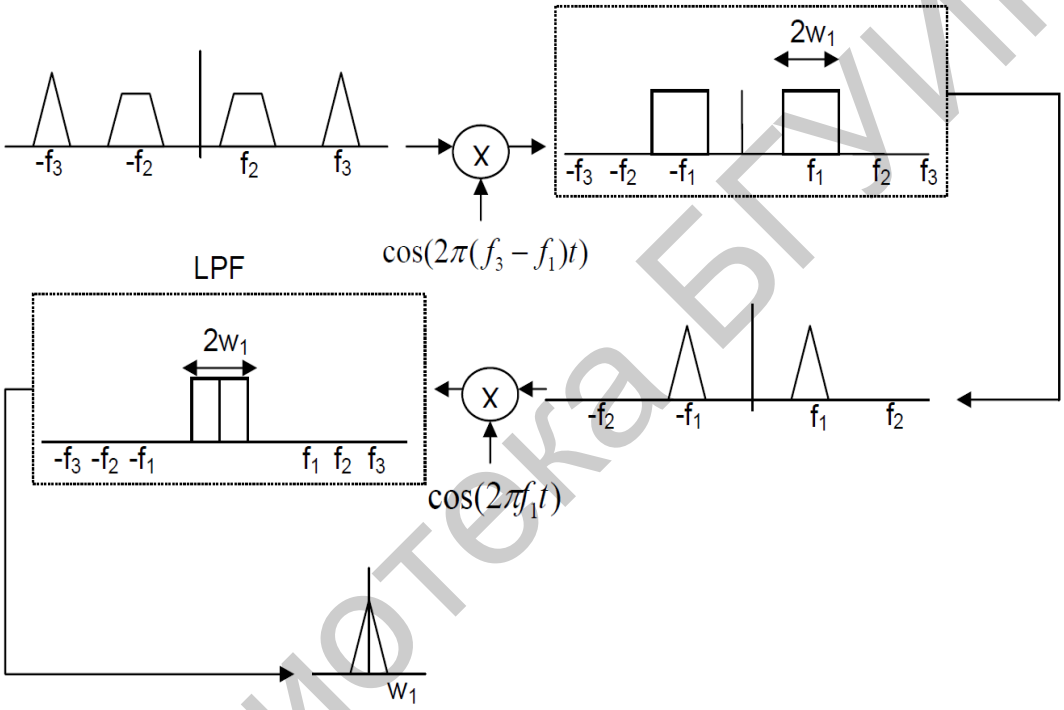


Рис. 23. Процедура выделения сигнала с треугольным спектром

1.5. Примеры построения временных диаграмм изучаемых сигналов. Амплитудная модуляция

Получение амплитудно-модулированных колебаний, вид которых иллюстрируется временными диаграммами, приведенными на рис. 24.

```
>> t=0:0.01:5;
>> am1=1;
>> ac=1;
>> m=am1*cos(2*pi*0.5*t);
>> c=ac*cos(2*pi*10*t);
>> y=(1+m).*c;
>> subplot(3,1,1);
>> plot(t,m);
>> subplot(3,1,2);
```

```
>> plot(t,c);
>> subplot(3,1,3);
>> subplot(t,y);
>> plot(t,y);
```

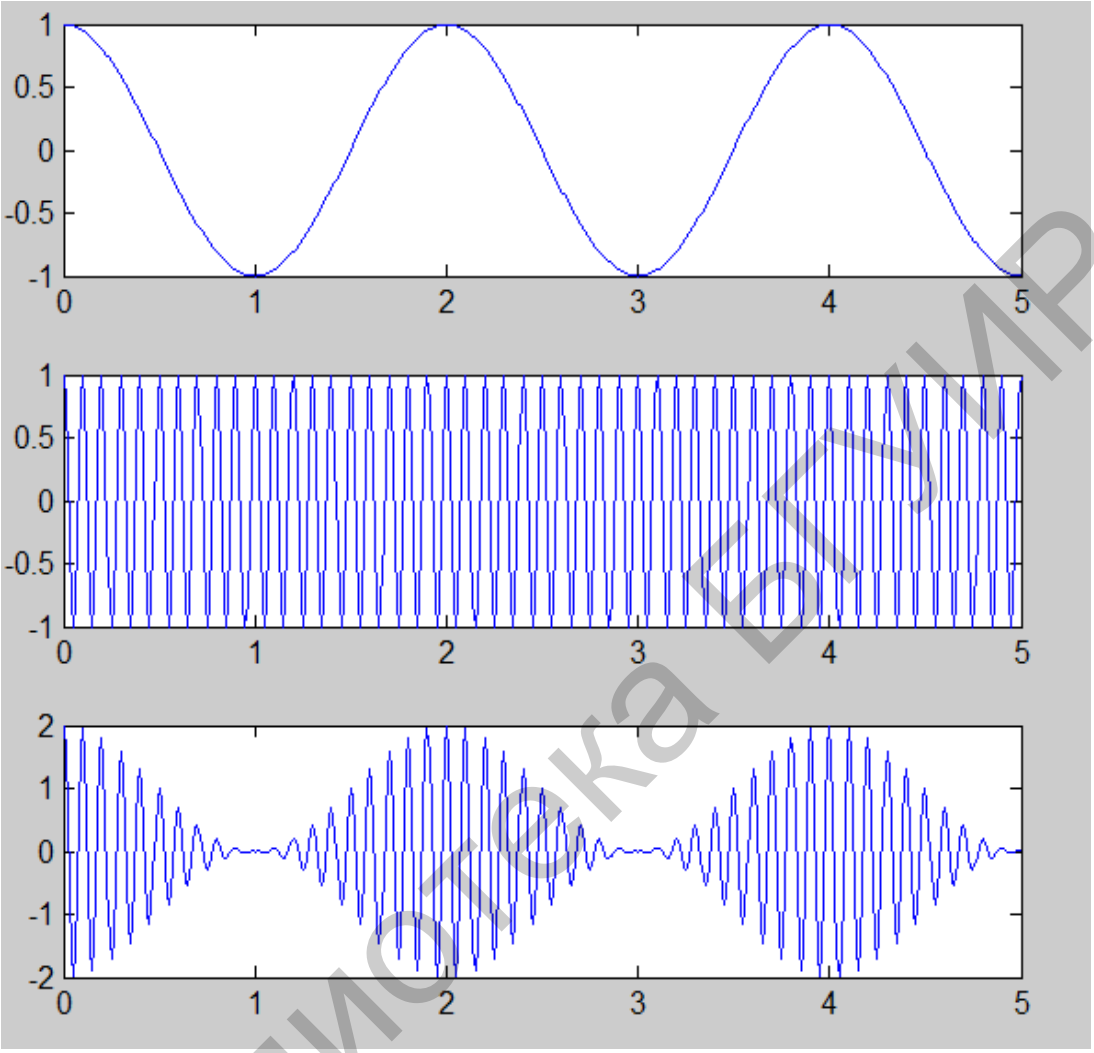


Рис. 24. Временные диаграммы, поясняющие процесс амплитудной модуляции

Частотная модуляция

Коды для построения временной диаграммы частотно-модулированного колебания, временная диаграмма которого показана на рис. 25.

```
>> t=0:10*10^(-6):0.001;
>> fm1=10^3;
>> x=2*sin(2*pi*fm1*t);
>> fc=20*10^6;
>> fs=100*10^6;
>> dph=8*10^6;
>> ini_phase=0;
>> y=fmmod(x,fc,fs,dph,ini_phase);
>> subplot(1,2,1);
```

```
>> plot(x);
>> title('message signal');
>> subplot(1,2,2);
>> plot(y);
>> title('Freqency modulation');
>>
```

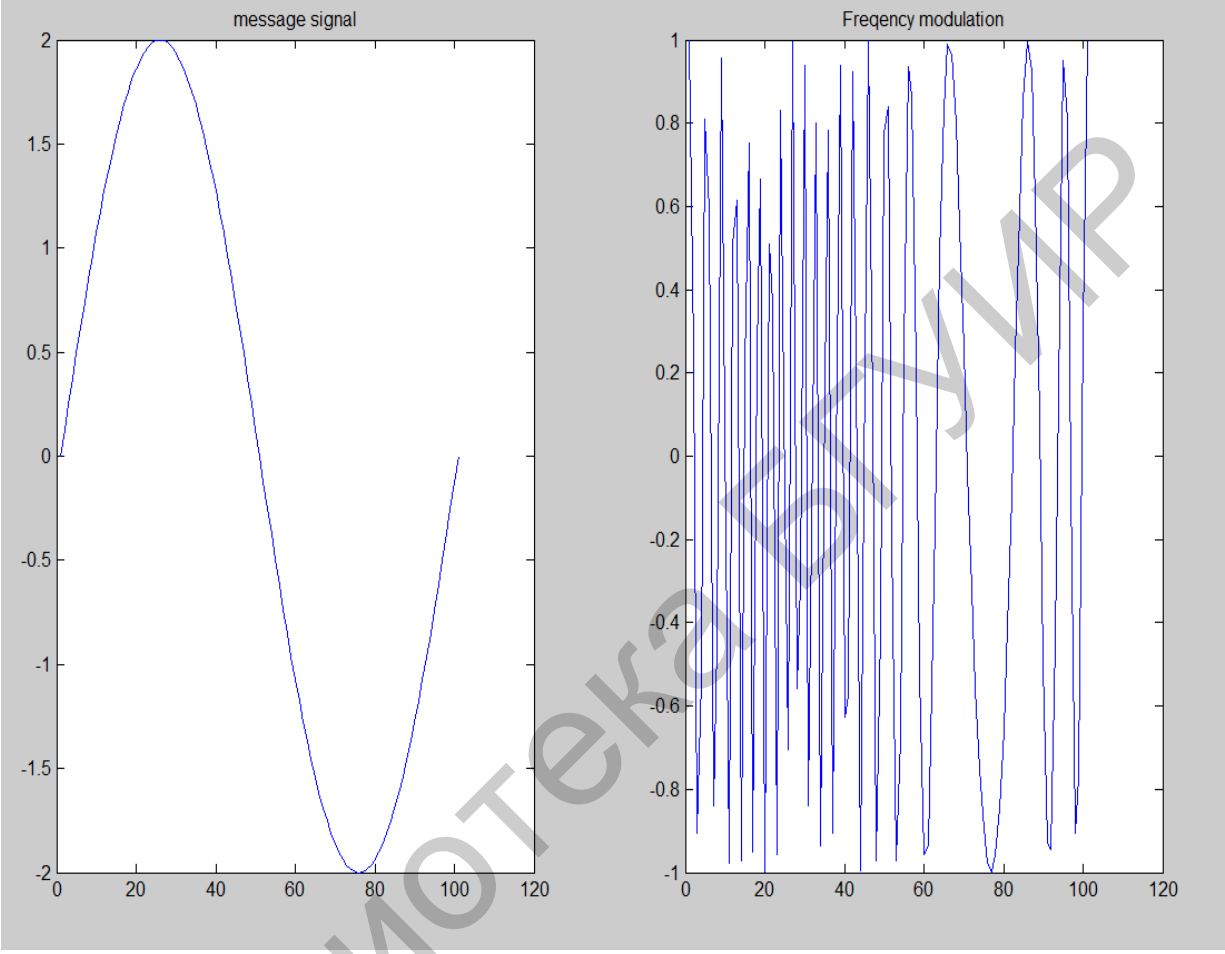


Рис. 25. Временная диаграмма частотно-модулированного колебания

Фазовая модуляция

Коды для построения временной диаграммы фазомодулированного колебания, временная диаграмма которого показана на рис. 27.

```
>> t=0:10*10^(-6):0.001;
>> fm1=10^3;
>> x=2*sin(2*pi*fm1*t);
>> fc=20*10^6;
>> fs=100*10^6;
>> dph=pi/2;
>> ini_phase=0;
>> e=pmmod(x,fc,fs,dph,ini_phase);
>> y=pmmod(x,fc,fs,dph,ini_phase);
>> subplot(1,2,1);
>> plot(x);
```

```
>> title('message signal');
>> subplot(1,2,2);
>> plot(y);
>> title('phase modulation');
```

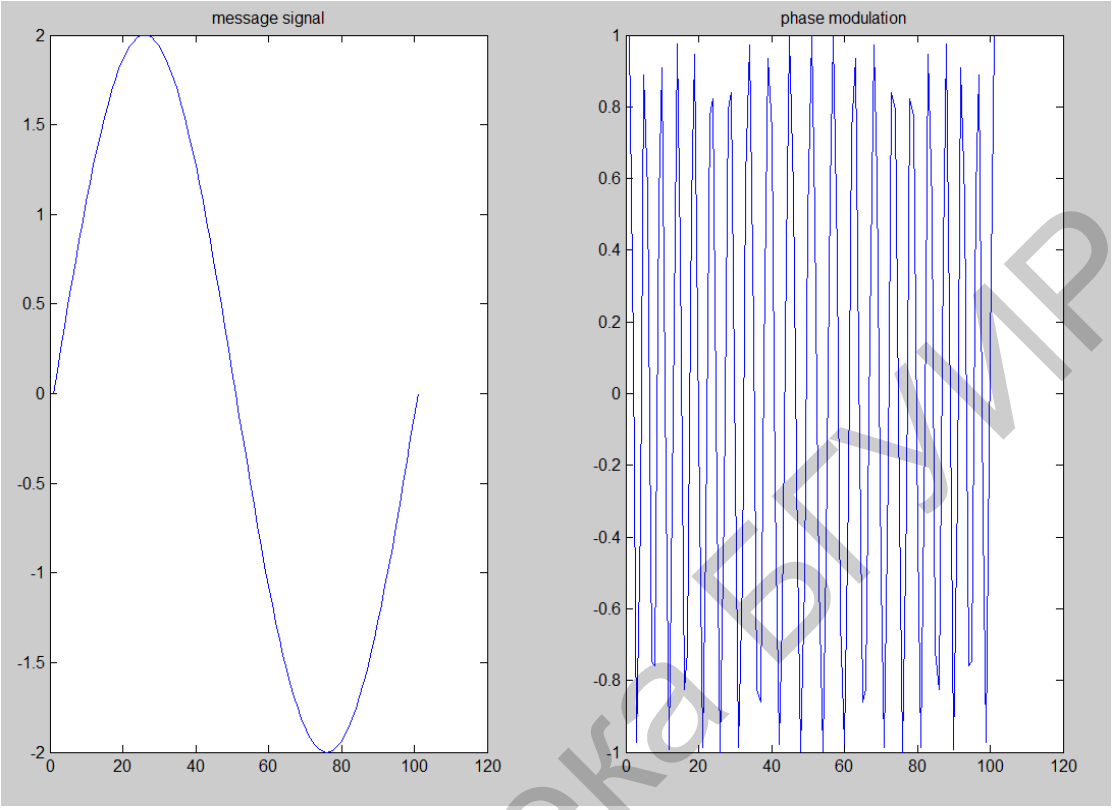


Рис. 26. Временная диаграмма фазомодулированного колебания

2. Задание на выполнение лабораторной работы

Задание. Написать программу, которая будет иллюстрировать процесс передачи трех информационных сигналов по каналу связи с FDM. В качестве информационных сигналов выбрать:

- амплитудно-модулированный сигнал с частотой управляющего сигнала $F = 4$ кГц, несущей частотой $f = 40$ кГц, коэффициентом модуляции m=75 %;
- частотно модулированный сигнал с частотой управляющего сигнала, равной 10 кГц, несущей частотой $f = 1000$ кГц, девиацией частоты $\Delta f = 50$ кГц;
- фазомодулированный сигнал с частотой управляющего сигнала $F = 10$ кГц, несущей частотой $f = 2000$ кГц, индексом фазовой модуляции $\beta = 10$.

Построить временные и спектральные диаграммы этих сигналов, считая, что амплитуда несущих у всех сигналов равна 1 В.

3. Контрольные вопросы

1. В чем состоит суть процесса модуляции?
2. Что такое процесс демодуляции?

3. Изобразите структурную схему модулятора в системе MATLAB.

4. Покажите, как можно представить во временной области демодулированный сигнал.

5. Дайте спектральное представление демодулировнного сигнала.

6. Каким разностным уравнением можно описать работу демодулятора?

7. Обоснуйте введение фильтрации при выполнении демодуляции в системе MATLAB/SIMULINK.

8. Каким образом создается фильтр низких частот в системе MATLAB/SIMULINK?

9. В чем состоит принцип работы технологии частотного разделения каналов?

10. Чем отличается принцип нелинейной модуляции от принципа параметрической модуляции?

Лабораторная работа №3
ИЗУЧЕНИЕ ДЕЛЬТА-МОДУЛЯЦИИ И СИГМА-ДЕЛЬТА МОДУЛЯЦИИ

1. Краткие теоретические сведения

В аналоговых системах связи помимо АМ-, ЧМ- и ФМ- технологий используются еще технологии импульсной модуляции, такие как Pulse Amplitude Modulation (PAM), Pulse Density Modulation (PDM), Pulse Time Modulation (PTM), Pulse Width Modulation (PWM), известная также как Pulse Duration Modulation (PDM), имеющие ряд разновидностей.

PAM – простейший вид импульсной модуляции, так как характеризуется как простотой получения модулированных колебаний, так и их детектирования. Однако этому виду модуляции присущ недостаток – он не допускает амплитудного ограничения. В связи с этим PAM мало пригодна в тех случаях, когда одним из главных требований является повышение помехоустойчивости канала. Поэтому PAM находит применение в качестве промежуточного преобразования при осуществлении, а также при детектировании более сложных видов амплитудной модуляции, например, PTM или PWM. На рис. 1 показан процесс формирования сигнала амплитудной импульсной модуляции (АИМ), в котором сначала из последовательности $s_p(t)$ прямоугольных импульсов постоянной амплитуды и длительности τ, следующих с периодом повторения T, формируется последовательность $s(t)$ модулированных импульсов, которыми затем будет модулироваться несущая частота радиосигнала. Сигнал сообщения показан пунктирной линией. При этом в качестве несущего колебания используется последовательность немодулированных импульсов, а информационным параметром является амплитуда модулированных импульсов в соответствии с передаваемым сообщением.

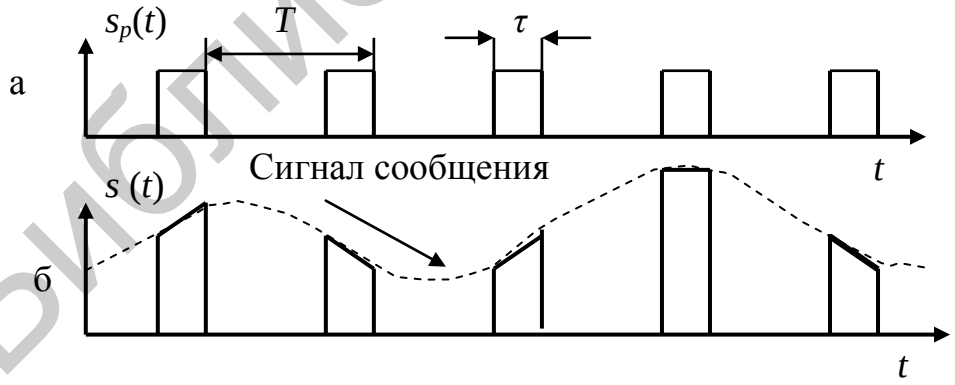


Рис. 1. Преобразование последовательности $s_p(t)$ (а) в последовательность $s(t)$ модулированных импульсов (б)

При модуляции PWM импульсы постоянной амплитуды изменяют свою длительность пропорционально амплитуде сигнала сообщения. При этом мак-

симальной амплитуде аналогового сигнала соответствует наиболее широкий импульс, а минимальная амплитуда сигнала сообщения создает импульс самой малой длительности.

При модуляции PPM положение импульсов, постоянных по ширине внутри предписанного промежутка времени, изменяется в соответствии с амплитудой отсчета аналогового сигнала сообщения: чем выше амплитуда отсчета, тем дальше вправо сдвигается импульс внутри предписанного временного промежутка.

Возможны три типа PWM, когда в зависимости от величины аналогового сигнала:

- центр импульса фиксируется и расстояние между передним и задним фронтами импульса расширяется или сокращается;
- передний фронт импульса удерживается на месте и относительно его положения меняется положение его заднего фронта;
- задний фронт фиксируется и относительно его положения изменяется положение переднего фронта.

Pulse Position Modulation (PPM) – это модуляция, в которой *M* битов сообщения кодируются передачей одного импульса с определенным сдвигом по оси времени. Модуляция повторяется каждые *T* с, так что скорость передачи определяется отношением *M/T*. Первоначально PPM использовалась в оптических системах связи, где практически отсутствуют искажения вследствие многолучевого распространения сигналов.

1.1. Импульсная кодовая модуляция

В ретроспективе появление PCM, подобно другим великим открытиям, представляется простым и очевидным. Ранними причинами дискретизации сигнала было переплетение отсчетов от различных источников телеграфных сигналов, чтобы обеспечить их передачу по одному телеграфному кабелю. Телеграфное мультиплексирование (time-division multiplex, TDM) использовал американский изобретатель M. B. Farmer еще в 1853 году. Инженер по электротехнике W. M. Miner в 1903 году применил электромеханический коммутатор для мультиплексирования телеграфных сигналов в телефонной связи, получив понятную передачу речевого сигнала при скорости дискретизации 3500…4300 Гц.

PCM изобрел британский инженер Alec Reeves в 1937 году во время работы в ITT (International Telephone and Telegraph) во Франции. Он получил патент Франции в 1938 году и патент США в 1943 году.

Слово pulse в термине PCM иногда вводит в заблуждение, так как по сути в этом термине никаких импульсов нет. Появление слова pulse – это естественное следствие развития техники PCM по двум направлениям: Pulse Width Modulation (PWM) и Pulse Position Modulation (PPM), в которых информация кодируется в виде бинарных импульсных сигналов, изменявших свою длительность и положение соответственно в зависимости, например, от амплитуды аналогового модулирующего сигнала. В этом смысле PCM не имеет никакого сходства с

этими обоими методами кодирования сигнала, за исключением того, что бинарные числа PCM представляются электрическими импульсами.

PWM можно рассматривать как разновидность Pulse Density Modulation (PDM), которая также используется для представления аналогового сигнала в цифровой области. В PDM специфическое значение амплитуды аналогового сигнала не кодируется в импульсы так, как это делается в PCM. В PDM вычисляется относительная плотность импульсов, что соответствует амплитуде аналогового сигнала. В PDM битовый поток кодируется с помощью процесса сигма-дельта модуляции, который использует однобитный квантователь, в результате получается либо единичный, либо нулевой бит в зависимости от знака изменения мгновенного значения аналогового сигнала. Единичный бит соответствует увеличению амплитуды аналогового сигнала, а нулевой бит – уменьшению. Разница между единичным или нулевым битами и мгновенным значением аналогового сигнала представляет собой ошибку квантования, которая поступает в цепь обратной отрицательной связи сигма-дельта модулятора, где она усредняется за счет применения интегратора.

На рис. 2 показано распределение избыточной информации при квантовании сигнала в PCM.

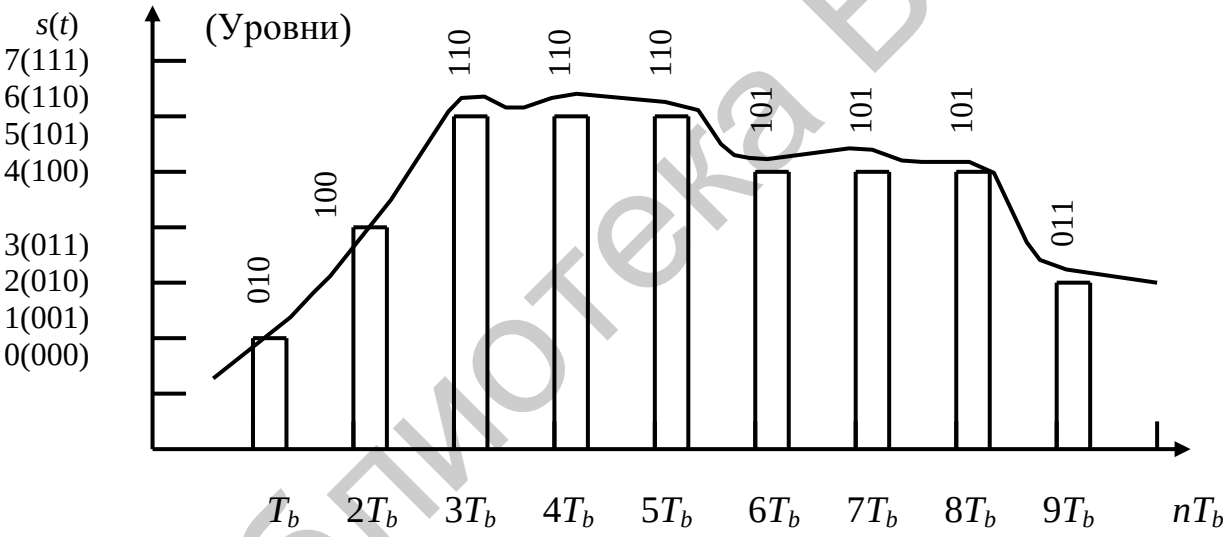


Рис. 2. Избыточная информация в PCM

Необходимость наличия предсказателя при PCM:

- когда выбрана скорость дискретизации как множество отсчетов, которое генерируется при скорости Найквиста, то может генерироваться неприемлемый чрезмерный шум квантования (в некоторых случаях сигнал может сильно изменяться, так как изменения могут следовать от максимального до минимального уровня сигнала);

- если увеличить скорость дискретизации, шум квантования может уменьшиться, но битовая скорость DPCM (биты на отсчет, умноженные на скорость дискретизации) может отличаться (превосходить) от битовой скорости PCM.

1.2. Дифференциальная импульсно-кодовая модуляция

Дифференциальная импульсно-кодовая модуляция (Differential Pulse Code Modulation, DPCM) характеризуется следующими положениями:

- в PCM каждый отсчет сигнала кодируется независимо от всех других отсчетов;

- средние изменения в амплитудах между последовательными отсчетами относительно невелики;

- следовательно, схема кодирования, что использует избыточность в отсчетах, будет проявляться в более низкой битовой скорости для источника выходного сигнала;

- в PCM кодируется разность между последовательными отсчетами лучше, чем сами отсчеты;

-так как разница между отсчетами ожидается значительно меньшей, чем действительные амплитуды отсчетов, то всего несколько битов требуется, чтобы представить эту разницу;

- в этом случае квантуется и передается по каналу связи последовательность разностей сигналов $e(n) = s(n) - s(n\text{-}1)$, где $s(n)$ – последовательные отсчеты сигнала $s(t)$.

Структурные схемы кодера и декодера DPCM показаны на рис. 3.

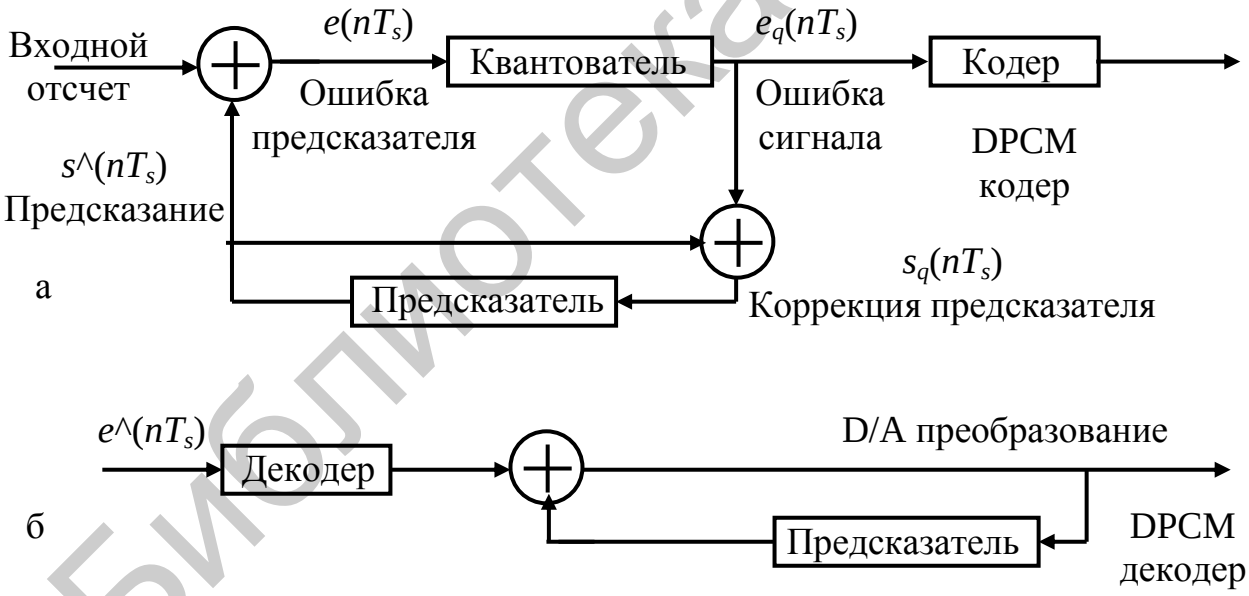


Рис. 3. Структурные схемы кодера (а) и декодера (б) DPCM

1.3. Логарифмическая импульсно-кодовая модуляция.

Причины обращения к логарифмической импульсно-кодовой модуляции

В линейном и равномерном квантователе ошибка квантования отсчета составляет

$$e_k = x(t) - x_q(kT_s) \qquad (1)$$

и максимальная величина ошибки в оценке амплитуды отсчета равна

$$Max|e_k| = \delta/2. \qquad (2)$$

Так, если $x(t)$ мало по амплитуде и также малые амплитуды более вероятны для входного сигнала, чем амплитуды, наблюдающиеся вблизи максимальных положительных и отрицательных значений сигнала $\pm V$, то можно предположить, что шум квантования такого входного сигнала будет значительным по сравнению с мощностью сигнала $x(t)$. Это предполагает, что SQNR низкого по уровню сигнала будет невысоким и потому и неприемлемым. В практических реализациях импульсной кодовой модуляции (ИКМ) часто стараются разрабатывать квантователи так, чтобы SQNR почти не зависел от амплитуды искажений аналогового входного сигнала $x(t)$.

Это достигается использованием неравномерного квантователя, который обеспечивает малые ошибки квантования для малых амплитуд входного сигнала и при относительно большем размере шага квантования, когда амплитуда сигнала большая. Передаточная характеристика такого квантователя показана на рис. 4. Неравномерный квантователь можно рассматривать, как эквивалентное устройство для выполнения процесса амплитудных предыскажений, за которым следует процесс равномерного квантования с фиксированным размером шага квантования δ. Кратко обсудим форму желательной характеристики функции предыскажения и сжатия $y = c(x)$ для неравномерного квантования, показанной на рис. 5.

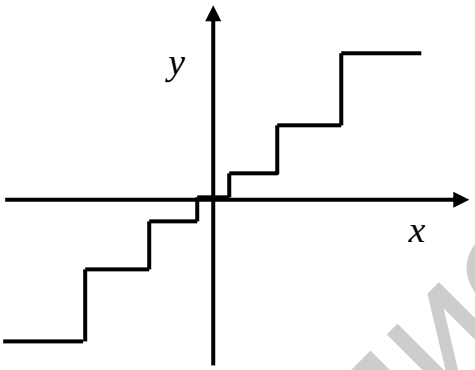


Рис. 4. Характеристика вход-выход неравномерного квантователя

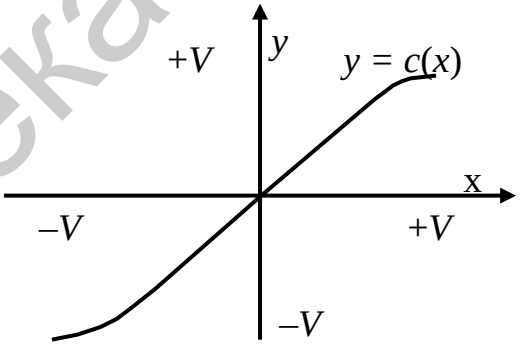


Рис. 5. Желательная передаточная характеристика неравномерного квантователя

Математически $y = c(x)$ – это монотонно возрастающая функция x с нечетной симметрией. Монотонность этой функции обеспечивает то обстоятельство, что существует за пределами области $x(t)$ такая функция $c^{-1}(x)$ и является уникальной по отношению к функции $c(x)$ в том смысле, что справедливо соотношение

$$c(x)\, c^{-1}(x) = 1.$$

Напомним, что операция $c^{-1}(x)$ нужна в ИКМ декодере. Чтобы обратно получить исходный сигнал неискаженным. Свойство нечетной симметрии, т. е. $c(-x) = c(x)$, просто принимает во внимание полный диапазон изменения $x(t)$ в пределах $\pm V$ и в дальнейшем предполагает, что

$$c(x)=\begin{cases}+V, & x=+V;\\ 0, & x=0;\\ -V, & x=-V.\end{cases} \qquad (3)$$

Пусть на k-м шаге после нелинейного квантования шаг квантователя будет δ и число интервалов квантования равно M. Пусть k-й уровень представления амплитуды входного сигнала после квантования, когда уровень входного сигнала находится между x_k и x_{k+1}, равен y_k, где

$$y_k = 0{,}5(x_k + x_{k+1}),\ k = 1, 2, \ldots, M-1. \qquad (4)$$

Для ошибки квантования e_k имеем

$$e_k = x - y_k,\ x_k < x < x_{k+1}.$$

Из соотношения (3) видно, что текущий размер шага квантования δ_k будет малым, если будет мала скорость изменения функции $c(x)$, т. е. $dc(x)/dx$. Это проявляется в изменении наклона функции $y = c(x)$. Поэтому примем следующее упрощение:

$$\frac{dc(x)}{dx}=\frac{2V}{M}\frac{1}{\delta_k},\ k=0,1,\ldots,M-1 \qquad .(5)$$

и

$$\delta_k = x_{k+1} - x_k,\ k=0,1,\ldots,M-1.$$

Отметим, что $2V/M$ – это фиксированный размер шага равномерного квантования квантователя, структурная схема которого приведена на рис. 6.

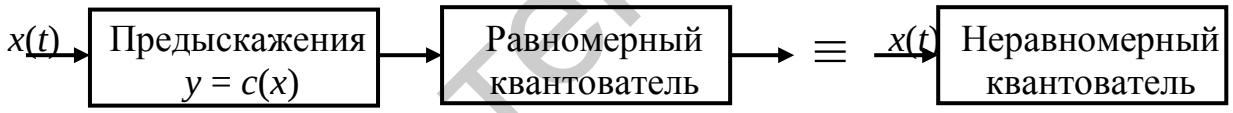


Рис. 6. Эквивалентная форма неравномерного квантователя

Допустим, что входной сигнал имеет нулевое среднее значение и его функция распределения вероятностей $p(x)$ симметрична относительно вертикальной оси координат. Далее для большого числа интервалов можно допустить, что в каждом интервале I_k, $k = 0, 1, \ldots, M-1$ функция $p(x)$ постоянна. Поэтому если значение входного сигнала x(t) находится между x_k и x_{k+1}, т. е. $x_k < x \le x_{k+1}$, то тогда справедливо равенство $p(x) = p(y_k)$.

Поэтому вероятность I_k того, что x лежит в k-м интервале,

$$I_k = p_k = P(x_k < x \le x_{k+1}) = p(y_k)\delta_k, \qquad (6)$$

где

$$\sum_{k=0}^{M-1} P_r(x_k < x \le x_{k+1}) = 1.$$

Среднеквадратичную ошибку квантования определим следующим образом:

$$\bar{e^2} = \int_{-V}^{+V} (x-y_k)^2 p(x)dx = \sum_{k=0}^{M-1}\int_{x_k}^{x_{k+1}} (x-y_k)^2 p(y_k)dx = \sum_{k=0}^{M-1}\frac{p_k}{\delta_k}\int_{x_k}^{x_{k+1}} (x-y_k)^2 dx =$$

$$= \sum_{k=0}^{M-1} \frac{p_k}{\delta_k} \frac{1}{3}\left[(x_{k+1} - y_k)^3 - (x_k - y_k)^3 \right] =$$

$$= \sum_{k=0}^{M-1} \frac{1}{3}\left(\frac{p_k}{\delta_k} \right) \left\{ \left[x_{k+1} - \frac{1}{2}\left(x_k + x_{k+1} \right) \right]^2 - \left[x_k - \frac{1}{2}\left(x_k + x_{k+1} \right) \right]^2 \right\} =$$

$$= \frac{1}{3} \sum_{k=0}^{M-1} \frac{p_k}{\delta_k} \frac{1}{4} \delta_k^3 = \sum_{k=0}^{M-1} p_k \delta_k^3 . \qquad (7)$$

Подставим

$$\delta_k = \frac{2V}{M} \left[\frac{dc(x)}{dx} \right]^{-1}$$

в полученное выражение и найдем приближенное значение среднеквадратичной ошибки в виде

$$e^{\bar{2}} = \frac{V^2}{3M^2} \int_{-V}^{+V} p(x) \left[\frac{dc(x)}{dx} \right]^{-2} dx . \qquad (8)$$

Полученное выражение предполагает, что среднеквадратичная ошибка, обусловленная неравномерным квантованием, может быть выражена в терминах непрерывной переменной x, $-V < x < +V$, которая имеет плотность распределения вероятностей $p(x)$:

$$e^{\bar{2}} = \frac{V^2}{3M^2} \int_{-V}^{+V} p(x) \left[\frac{dc(x)}{dx} \right]^{-2} dx . \qquad (9)$$

Теперь имеем выражение для SQNR неравномерного квантователя:

$$SQNR = \left(\frac{3M^2}{V^2} \right) \frac{\int_{-V}^{+V} x^2 p(x) dx}{\int_{-V}^{+V} p(x) \left[\frac{dc(x)}{dx} \right]^{-2} dx} . \qquad (10)$$

Полученное выражение является важным в том смысле, что дает ключ к желаемой форме функции сжатия $y = c(x)$, такой, что SQNR можно сделать независимой от распределения вероятностей сигнала $x(t)$. Легко видеть, что желаемое условие состоит в том, чтобы $\frac{dc(x)}{dx} = \frac{K}{x}$, где –V < x < +V и K – положительная константа, т. е.

$$c(x) = V + K \ln\left(\frac{x}{V} \right) \quad для \quad x > 0 \qquad (11)$$

и

$$c(x) = -c(x). \qquad (12)$$

З а м е ч а н и е. Если $c(x) \to \pm\infty$, тогда $x \to 0$ с другой стороны. Следовательно, это $c(x)$ нереализуемо на практике. Кроме того, функция сжатия $c(x)$

должна проходить через начало координат, т. е. $c(x) = 0$ для $x = 0$. Эти обстоятельства предопределяют использование функции сжатия в практических системах.

Существуют два популярных стандарта для нелинейного квантования, известные как

- μ-закон сжатия;
- А-закон сжатия.

Первый закон популярен в США, Японии, Канаде и некоторых других странах, в то время как второй закон используется в Европе и большинстве других стран, включая Индию, придерживающихся стандартов ITU-T.

Функция сжатия $c(x)$ для μ-закона сжатия

$$\frac{c(|x|)}{V} = \frac{\ln\left(1+\frac{\mu|x|}{V}\right)}{\ln(1+\mu)},\ 0 \le \frac{|x|}{V} \le 1{,}0, \qquad (13)$$

μ – константа здесь. Типичные значения μ лежат в пределах от 0 до 255. При $\mu = 0$ имеем дело с линейным, т. е. равномерным квантованием.

Функция сжатия А-закона служит целям сжатия и расширения и имеет вид (рис. 6)

$$\frac{c(|x|)}{V} = \begin{cases} \dfrac{A\frac{|x|}{V}}{1+\ln A},\ 0 \le \dfrac{|x|}{V} \le \dfrac{1}{A}; \\ \dfrac{1+\ln\left(A\frac{|x|}{A}\right)}{1+\ln A},\ \dfrac{1}{A} \le \dfrac{|x|}{A} \le 1{,}0. \end{cases} \qquad (14)$$

Здесь параметр A представляет собой константу, типичное значение которой, что широко используется на практике, равно 87,5.

Для телефонных каналов с 8 битами на отсчет дискретизации и 8 килобит/с типичное значение SQNR = 38,4 дБ, что достигается на практике.

Приближение логарифмической функции сжатия используется для равномерного квантования, а схемы с импульсно-кодовой модуляцией, использующие неравномерное квантование, называются еще схемами логарифмической импульсно-кодовой модуляции, или Log PCM.

Пример 1. Постройте функцию сжатия $c(x)$ для $\mu = 50$ и $V = 2{,}0$ *В*.

Пример 2. Постройте функцию сжатия $c(x)$ для A-закона сжатия-расширения, когда $A = 50$ и $V = 2{,}0$ *В*.

Пример 3. Прокомментируйте эффективность нелинейного (неравномерного) квантования, когда амплитудное значение сигнала известно и значительно ниже, чем максимально допустимое напряжение V.

1.4. Дельта-модуляция

Дельта-модуляция (delta modulation, DM) – это технология преобразования аналогового сигнала в цифровую форму и может рассматриваться как простейшая форма DPCM. Но есть некоторые различия между этими двумя технологиями. В DPCM-технологии последовательные отсчеты кодируются в поток *n*-битовых данных. Однако в DM переданные данные укладываются в однобитовый поток их передачи.

Впервые дельта-модуляцию предложили сотрудники лаборатории ITT во Франции E. M. Delotaine, S.Van Mierlo и B. Derjavitch в 1946 году. Принципы ее получения были снова открыты в Голландии несколькими годами позже. В 1950 году C.C.Cutler (сотрудник лаборатории Bell Telephone Labs в США) получил важный патент на дифференциальную PCM, в котором он раскрыл существенные особенности концепции сигма-дельта модуляции.

Характерные особенности DM:

- аналоговый сигнал представляется в виде серии сегментов с квантованными величинами их амплитуд;
- для получения информации об увеличении или уменьшении амплитуды каждого последующего сегмента относительно предыдущего каждый раз сравнивается амплитуда сегмента аппроксимированного сигнала с исходным значением аппроксимируемого сигнала;
- в результате сравнения значений оригинального и аппроксимированного сигналов получается информации, необходимая для установки единичного или нулевого бита в зависимости от результата этого сравнения.;
- только изменение информации передается по каналу связи, т. е. только увеличение или уменьшение амплитуды сигнала по отношению к предыдущему отсчету передается по каналу, а если никаких изменений не происходит, то передается нулевой или единичный бит, соответствующий состоянию предыдущего сегмента;
- используя технологию DM, можно получить высокие значения отношения сигнал/шум, т. е. сигнал дискретизируется со скоростью, превышающей скорость Найквиста.

Математическое описание имеет вид

$$e(kT_s)=s(kT_s) - s\hat{}(kT_s),$$

где $e(kT_s)$ – ошибка предсказания; $s(kT_s)$ – неквантованный входной отсчет, $s\hat{}(kT_s)$ – уровень предсказанного сигнала.

На вход предсказателя поступает сигнал, описываемый выражением

$$s_q(kT_s)=s\hat{}(kT_s) + e_q(kT_s),$$

где $e_q(kT_s)$ – сигнал ошибки квантования.

На выходе квантователя формируется суммарный сигнал

$$e_q(nT_s) = e(nT_s) + q(nT_s),$$

где $q(nT_s)$ – ошибка квантования.

Таким образом, DM – это DPCM-схема, в которой разница сигналов кодируется просто в 1 бит и имеет следующие особенности:

- DM может рассматриваться как варианты расширения DPCM и PCM;

- DM оказывается лучше, чем PCM и DPCM по полосе частот или требованиям к скорости передачи данных, так как имеет большую емкость передаваемой информации с использованием нескольких бит;

- схема DM посылает только разницу между амплитудами импульсов: если импульс в момент времени t_{n+1} превышает амплитудное значение импульса в момент времени t_n, тогда соответствующий этой ситуации бит принимает, скажем, единичное значение, используемое для указания на положительную тенденцию изменения амплитуд в последовательности отсчетов;

- если амплитуда имеет более низкое значение, чем величина амплитуды предшествующего импульса, тогда формируется отрицательный бит, указывающий на отрицательную тенденцию в формировании амплитуд последовательности анализируемых отсчетов;

- схема DM особенно хорошо работает при малых изменениях амплитуд отсчетов в их последовательности;

- если изменения амплитуд являются значительными в последовательности отсчетов, то будут формироваться очень большие величины ошибок.

Структурные схемы дельта-модулятора и демодулятора показаны на рис. 7 и 8 соответственно.

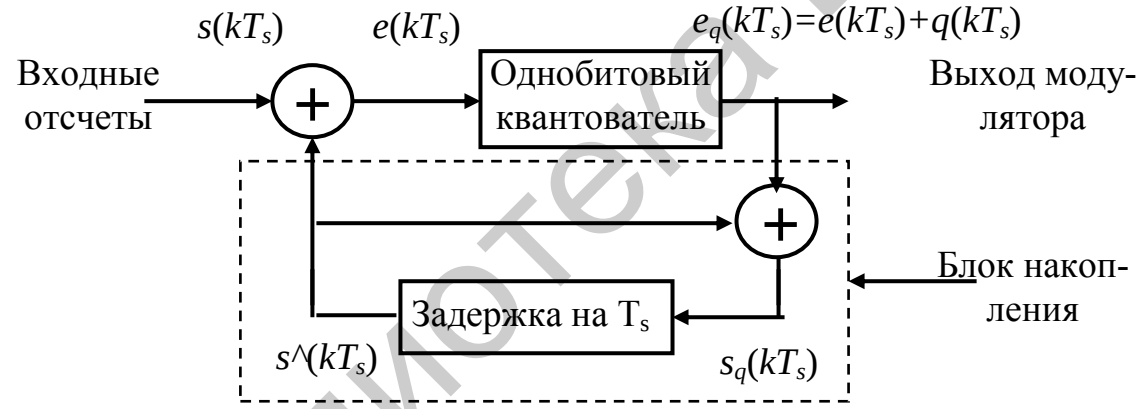


Рис. 7. Структурная схема дельта-модулятора

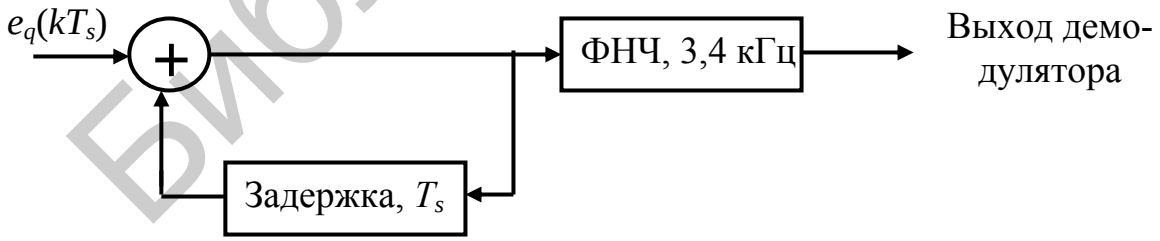


Рис. 8. Структурная схема демодулятора дельта-модуляции

Преимущества DM по сравнению с PCM и DPCM:

- скорость дискретизации и произведение пропускной способности канала на ширину его полосы пропускаемых частот очень мало;

- выполнение устройства передачи-приема много проще, так как не требуется преобразования аналогового сигнала в цифровую форму;

- вместо использования одного бита для указания положительной или отрицательной разности между текущим и предыдущим значениями сигнала можно использовать больше битов для квантования этой разницы;

- чем больше битов отведено для квантования уровней, тем выше точность выполнения преобразований.

Дельта-модуляция – это аналого-цифровое преобразование сигнала, в котором аналоговый сигнал аппроксимируется рядом прямоугольных сегментов. При этом:

- каждая величина амплитуды сегмента аппроксимированного сигнала сравнивается с уровнем исходного сигнала, чтобы определить, увеличивается или уменьшается амплитуда исходного сигнала на каждом последующем сегменте относительно его уровня на предыдущем сегменте;

- на выходе дельта-модулятора формируется импульс постоянной амплитуды и длительности, определяемой длительностью сегмента, т. е. увеличению или уменьшению амплитуды преобразуемого сигнала относительно предыдущего отсчета соответствует положительный или отрицательный сигнал на выходе дельта-модулятора.

Таким образом, при дельта-модуляции имеется только два уровня квантования и формируется только один бит на каждый отсчетный сегмент аппроксимированного аналогового сигнала.

Процесс дельта-модуляции сопровождается генерированием шума квантования двух типов, показанных на рис. 9:

Рис. 9. Дельта-модуляция:
аналоговый сигнал (кривая 1) и сигнал
квантователя линейной ДМ (кривая 2)

- granular noise (гранулированный шум);

- slope overload noise (шум перегрузки по крутизне).

Взаимосвязь между интенсивностью составляющих шума квантования следующая:

- с возрастанием длительности сегмента аппроксимации аналогового сигнала шум первого типа преобладает;

- при уменьшении длительности сегмента аппроксимации аналогового сигнала преобладает шум второго типа.

Чтобы смягчить влияние этих противоречивых тенденций, используется адаптивная дельта-модуляция.

1.5. Адаптивная дельта-модуляция

При адаптивной дельта-модуляции происходит автоматическое изменение длительности сегмента, аппроксимирующего непрерывный аналоговый сигнал при изменении последнего во времени: когда амплитуда непрерывного сигнала возрастает, длительность этого сегмента автоматически уменьшается и наоборот. Существует много методов реализации этой стратегии. Упомянем два наиболее широко использующихся в реализации адаптивной дельта-модуляции метода.

Метод 1. Размер шага адаптируется путем тестирования выходных импульсов дельта-модулятора: если их последовательность представляет строку импульсов одинаковой полярности, размер сегмента аппроксимирующего непрерывный аналоговый сигнал увеличивается до тех пор, пока не появится импульс противоположной полярности. После этого размер длительности сегмента уменьшается, и процесс начинает развиваться в том же направлении. Алгоритм можно пояснить с помощью табл. 1, где показано, как изменяется размер длительности сегмента, аппроксимирующего непрерывный сигнал в зависимости от последовательности сформированных на выходе модулятора импульсов.

Таблица 1

Алгоритм осуществления адаптивной дельта-модуляции

Последовательность импульсов на выходе дельта-модулятора	Число последовательно сформированных единичных или нулевых битов	Размер сегмента, аппроксимирующего аналоговый сигнал
X X X X 0 1	1	δ
X X X 0 1 1	2	δ
X X 0 1 1 1	3	2δ
X 0 1 1 1 1	4	4δ

X – в этом случае соответствует либо единичному, либо нулевому биту.

Метод 2 соответствует адаптивной дельта-модуляции, называемой еще continuosly variable slope delta-modulation (CVSDM), когда вместо подсчета числа последовательно сгенерированных импульсов одной полярности используется их интегрирование. Примером такого модулятора являются устройства MC34115, MC3418.

Естественно спросить, что лучше PCM или DM? Ответ зависит от критерия, используемого при сравнении и типа сообщения:

- при желании иметь относительно простые, дешевые системы нужно отдать предпочтение DM;
- чтобы получить более высокий уровень отношения сигнал/шум, PCM, вероятно, лучше;
- интерфейс существующего оборудования и совместимость устройств PCM имеют ряд преимуществ перед устройствами DM.

Итак, PCM и CVSDM – это дифференциальная техника квантования непрерывного сигнала. Оба вида модуляции работают с двумя уровнями квантования, т. е. формируют либо единичный, либо нулевой бит информации. CVSDM – это DM с адаптивным квантованием. Применение адаптивной техники к DM квантователю позволяет согласовать размер шага квантования со скоростью изменения входного аналогового сигнала. При этом согласовании декодер имеет возможность восстанавливать сигнал с малыми амплитудами без принесения в жертву издержек обработки сигналов с большой амплитудой.

Адаптивная DM характеризуется следующими особенностями:

- адаптивная DM – это модификация DM;

- ошибки квантования обусловлены шумом квантования из-за перегрузки по крутизне и гранулированного шума в результате адаптивного выбора шага квантования;

- таким образом, вариация размера шага в соответствии с этими принципами DM способна копировать изменения во входном сигнале.

Принципы, подчеркивающие все алгоритмы ADM в зависимости от выхода однобитного квантователя, сводятся к следующему:

- если выход однобитного квантователя высокий, т. е. единичный, размер шага квантования можно увеличить для следующего отсчета;

- если выход однобитного квантователя низкий, т. е. нулевой, то размер шага квантования можно уменьшить для обработки следующего отсчета;

- на участках с крутым подъемом сигнала s(t) размер шага увеличивается;

- если входной сигнал изменяется очень медленно, размер шага уменьшается.

Преимущества ADM:

- SNRQ улучшается по сравнению с DM;

- так как ADM имеет переменный размер шага квантования, его динамический диапазон шире, чем у DM.

Структурные схемы ADM кодера и декодера приведены на рис. 10, а, б.

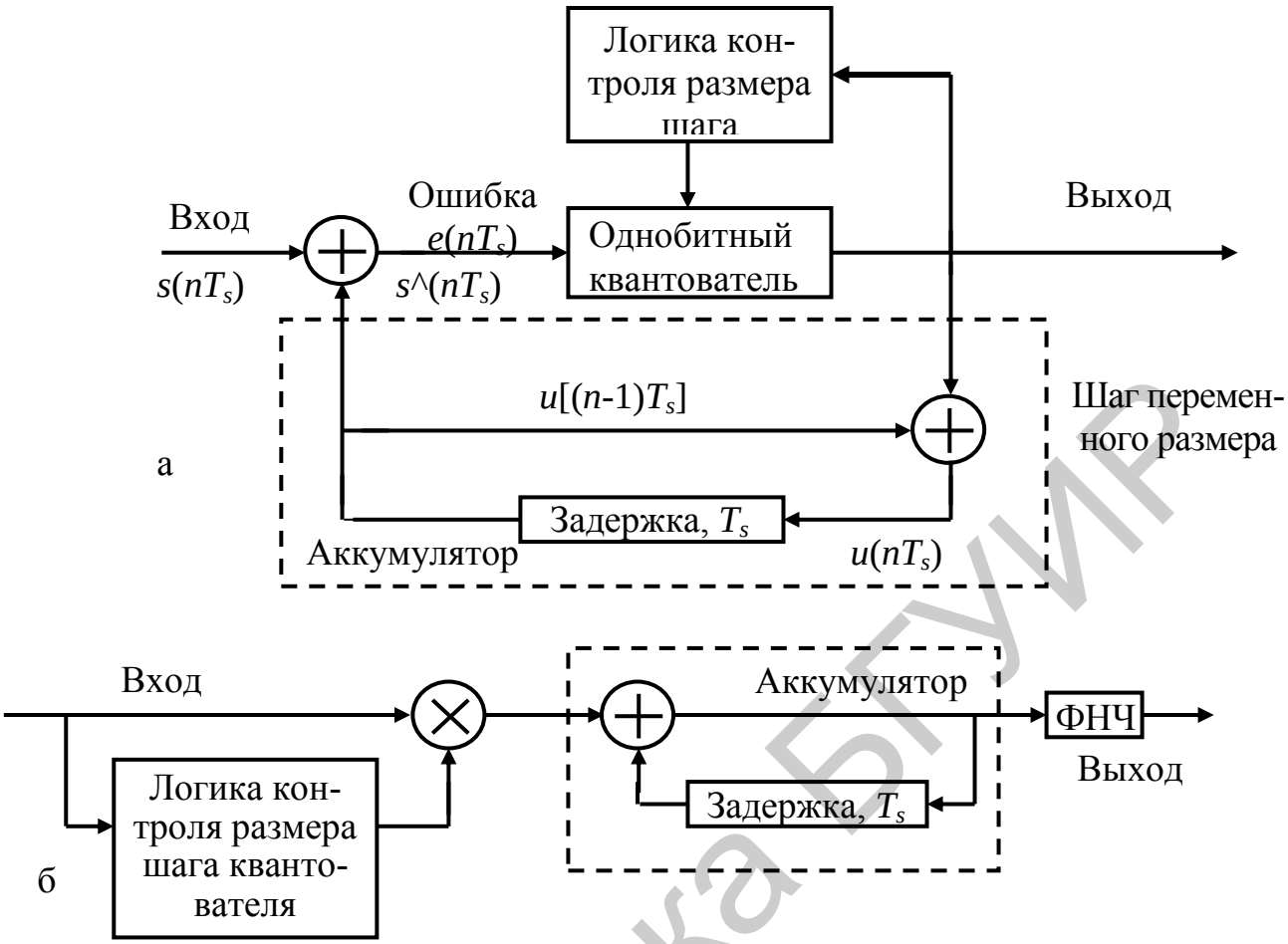


Рис. 10. Структурные схемы кодера (а) и декодера (б) ADM

Из рассмотрения рис. 11, на котором представлена форма ступенчатого сигнала, формируемого схемой ADM, видно, что на участках с крутым подъемом сигнала *s(t)* размер шага квантования увеличивается. Если входной сигнал изменяется очень медленно, размер шага уменьшается. Адаптивный амплитудный модулятор может также, продолжая уменьшение шага квантования, уменьшить расстояние между отсчетами.

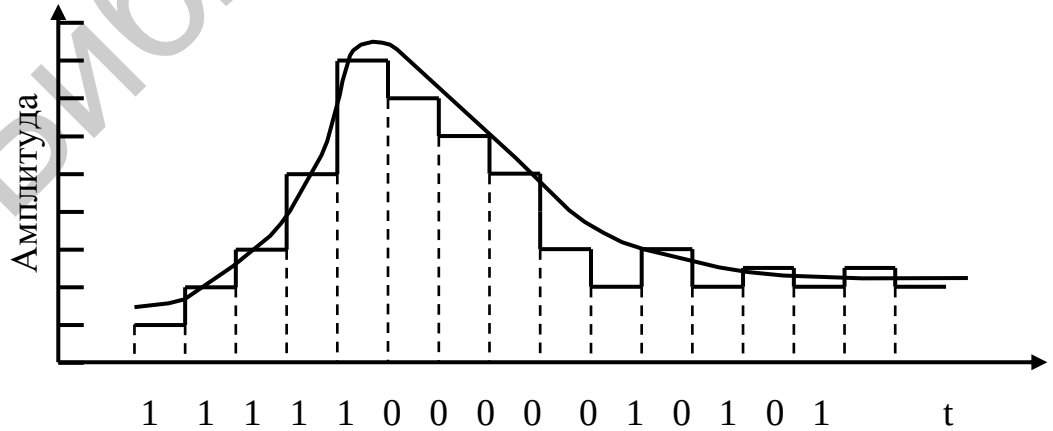


Рис. 11. Форма ступенчатого сигнала, формируемого схемой ADM, внизу показан выходной бинарный цифровой поток

1.6. Сигма-дельта модуляция

В 1954 году C.C.Cutler из Bell Labs получил патент на принцип передискретизации (oversampling) и формирование шума (noise shaping), имеющий цель получить высокую степень разрешения цифрового сигнала. Суть патента сводится к возможности создания АЦП, работающего на частоте Найквиста, но с передачей передискретизированного и со сформированным шумом сигнала без уменьшения скорости передачи данных. Таким образом, предложенный АЦП содержит все составляющие сигма-дельта модуляции, за исключением цифровой фильтрации и децимации, которые в то время были очень сложными и дорогими, так как использовали технологию ламповых приборов.

Случайные работы продолжались в течение нескольких лет, пока C.B. Brahm не получил патент в 1961 году, в котором он дал подробности устройства аналогового петлевого фильтра второго порядка в АЦП с многобитным формированием шума. Транзисторные схемы начали успешно заменять ламповые конструкции в это время, и это предложение открыло новые возможности для практического применения сигма-дельта архитектуры (рис. 11).

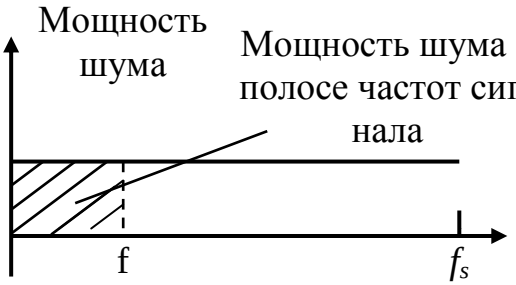


Рис. 12. Мощность шума в полосе частот сигнала

В 1962 году H. Inose, Y. Yasuda и J. Murakami детально разработали однобитную архитектуру с передискретизацией и формированием шума, предложенную C.C. Cutler' в 1954 году. Их конструкция уже использовала твердотельное устройство для изготовления сигма-дельта модуляторов первого и второго порядков. В 1963 году появилось прекрасное теоретическое обсуждение передискретизации и формирования шума. Правда, в этих статьях обсуждаемые устройства назывались дельта-сигма архитектурой, хотя название сигма-дельта появилось только в 1970 году, когда инженеры AT&T начали использовать термин «сигма-дельта модуляция». С этого времени используются оба термина, хотя термин «сигма-дельта модуляция» более корректен.

Итак, чтобы понять, как работает АЦП с сигма-дельта модулятором, достаточно понять суть передискретизации, формирования шума, цифровой фильтрации и децимации. Рассмотрим технику передискретизации при анализе в частотной области. При этом заметим, что преобразование постоянного уровня имеет ошибку квантования не более ½ LSB, а система дискретизованных данных характеризуется шумом квантования. При классическом *N*-битном квантовании АЦП имеет среднеквадратичное значение (rms) квантованного шума $q/\sqrt{12}$, если он предполагается равномерно распределенным в полосе частот от нулевой до $f_s/2$, где f_s – частота Найквиста, а q соответствует LSB, как это показано на рис. 12. Поэтому SNR с полностью используемой синусоидаль-

ной волной, представляющей входной сигнал, имеет величину $6{,}02 \cdot N + 1{,}76$ дБ. Если АЦП не использует своих возможностей по обработке входного сигнала полностью, тогда его эффективное разрешение описывается эффективной разрядностью (Effective Number Of Bits, ENOB) и оценивается выражением

$$ENOB = \frac{SNR - 1{,}76,\ дБ}{6{,}02,\ дБ}.$$

Если установить более высокую скорость дискретизации Kf_s, как это показано на рис. 13,б, тогда rms шума квантования останется прежним, т.е. равным $q/\sqrt{12}$, но шум распределится по всей новой полосе частот от 0 до $Kf_s/2$. Если затем использовать ФНЧ для фильтрации выходного сигнала АЦП, то устранится значительная часть шума квантования, но это никак не повлияет на полезный сигнал, т. е. его ENOB улучшится. В результате получится более высокое разрешение аналого-цифрового преобразования в АЦП с низким разрешением. Коэффициент K называется коэффициентом передискретизации. В этом заключены большие преимущества сигма-дельта модуляции.

Разрешение АЦП – минимальное изменение величины аналогового сигнала, которое может быть преобразовано данным АЦП – непосредственно связано с его разрядностью. В случае единичного измерения без учета шумов разрешение напрямую определяется разрядностью АЦП, которая характеризует количество дискретных значений, которые преобразователь может выдать на выходе. В двоичных АЦП разрядность измеряется в битах, в троичных АЦП – в тритах. Например, двоичный АЦП, способный выдать 256 дискретных значений, изменяющихся от 0 до 255, имеет разрядность 8 битов, поскольку $2^8 = 256$, троичный АЦП, имеющий разрядность 8 тритов, способен выдать 6561 дискретное значение, так как $3^8 = 6561$.

Разрешение по напряжению равно разности напряжений, соответствующих максимальному и минимальному выходному коду, деленной на количество выходных дискретных значений.

Пример 4. Диапазон входных значений 12-разрядного АЦП от –10 до 10 вольт. Оценить разрешение АЦП по напряжению.

Решение:

12-разрядный АЦП обеспечивает получение $2^{12} = 4096$ уровней квантования. Поэтому разрешение по напряжению двоичного АЦП равно $(10 - (-10))/4096 = 20/4096 = 0{,}00488$ В = 4,88 мВ.

На практике разрешение АЦП ограничено отношением сигнал/шум входного сигнала. При большой интенсивности шумов на входе АЦП различение соседних уровней входного сигнала становится невозможным, т. е. ухудшается разрешение. При этом реально достижимое разрешение описывается эффективной разрядностью ENOB, которая меньше, чем реальная разрядность АЦП. При преобразовании сильно зашумленного сигнала младшие разряды выходного кода практически бесполезны, так как содержат шум. Для достижения заявленной разрядности отношение сигнал/шум входного сигнала должно быть примерно 6 дБ на каждый бит разрядности.

Благодаря цифровой фильтрации скорость выходных данных снижается относительно скорости дискретизованных данных Kf_s, хотя и удовлетворяет критерию Найквиста. Это можно достичь передачей каждого M-го отсчета на выход АЦП и устранением всех остальных. Этот процесс называется децимацией с коэффициентом M. Несмотря на происхождение термина от латинского decet для числа 10, M может быть любым целым числом, обеспечивающим скорость данных на выходе большую, чем удвоенное значение полосы частот, занимаемой сигналом. Децимация нисколько не влияет на потери информации в процессе преобразования, как это видно из рис. 13, в.

Если используется передискретизация для улучшения разрешения, то нужно ее осуществлять с коэффициентом 2^N, чтобы получить N битов при увеличении разрешения. Сигма-дельта АЦП не нуждается в таком высоком коэффициенте передискретизации, потому что он не только ограничивает полосу частот сигнала, но также формирует шум квантования так, что большая его часть находится далеко за пределами полосы пропускания, как это видно из рис. 21, в.

Если взять однобитный АЦП (компаратор), подключить его вход к выходу интегратора, а выход – ко входу однобитного ЦАП, выход которого суммируется с входным аналоговым сигналом, то получим сигма-дельта модулятор, как это показано на рис. 14. Добавив цифровой ФНЧ и дециматор, получим сигма-дельта АЦП, ENOB которого будет выше, чем в том случае, как он ожидается только при коэффициенте передискретизации.

Интуитивно сигма-дельта АЦП работает следующим образом. Допустим, что на его вход поступает постоянное напряжение $+U_{BX}$. Интегратор формирует изменяющееся по треугольному закону напряжение в точке A схемы, приведенной на рис. 14. Выход компаратора через цепь обратной отрицательной связи с однобитным ЦАП суммируется в точке B. Предполагается, что усредненное напряжение на выходе ЦАП должно быть равно входному напряжению $+U_{BX}$. Среднее напряжение ЦАП контролируется однобитным потоком данных с выхода компаратора, имеющим однобитную плотность.

Если входной сигнал увеличивается до уровня $+U_{BXmax}$, число единиц в последовательном потоке данных увеличивается, а число нулей уменьшается. То же, но в обратном порядке произойдет при изменении U_{BX} до уровня $-U_{Bxmax}$, когда число единиц в битовом потоке данных уменьшится, а число нулей возрастет. Это простое описание показывает, что среднее значение входного напряжения содержится в последовательном потоке на выходе компаратора.

Цифровой фильтр и дециматор обрабатывают последовательный битовый поток и создают на выходе окончательный вид потока данных.

Для данного входного значения интервал между отдельными отсчетами данные, получаемые с выхода однобитного АЦП, в виртуальном смысле значения не имеют. Только когда усредняется большое число отсчетов, именно среднее значение их приобретает смысл. Сигма-дельта модулятор очень трудно анализировать во временной области, потому что напряжение на выходе в общем случае формируется случайным образом как однобитный выходной поток

данных. Если входной сигнал близок к $+U_{BXmax}$, то понятно, в сформированном выходном потоке битов преобладать будут биты единичные, а нулевых битов будет мало или их не будет вообще. Для сигналов в середине рабочего диапазона, изменяющегося от $-U_{BXmax}$ до $+U_{BXmax}$, число единичных и нулевых битов будет примерно одинаковым.

На рис. 15 показаны осциллограммы выходного напряжения интегратора для двух ситуаций. В ситуации, соответствующей рис. 15, а, на входе сигма-дельта АЦП действует напряжение, близкое нулевому уровню. Чтобы декодировать выход, нужно пропустить выходные отсчеты через цифровой ФНЧ, который в данном случае усредняет каждые 4 отсчета. На выходе формируется сигнал, равный 2/4. Это значение представляет бинарный нуль. Если усредняется большее количество отсчетов, то расширяется динамический диапазон обрабатываемых сигналов. Например, усреднение 4 битов дает разрешение в 2 бита, в то время как усреднение 8 отсчетов дает уже 4/8 или разрешение в 3 бита. На рис. 15, б показаны напряжения, соответствующие усреднению 4 битов, когда полученное среднее значение равно ¾ и среднее значение для 8 отсчетов получилось бы равным 6/8.

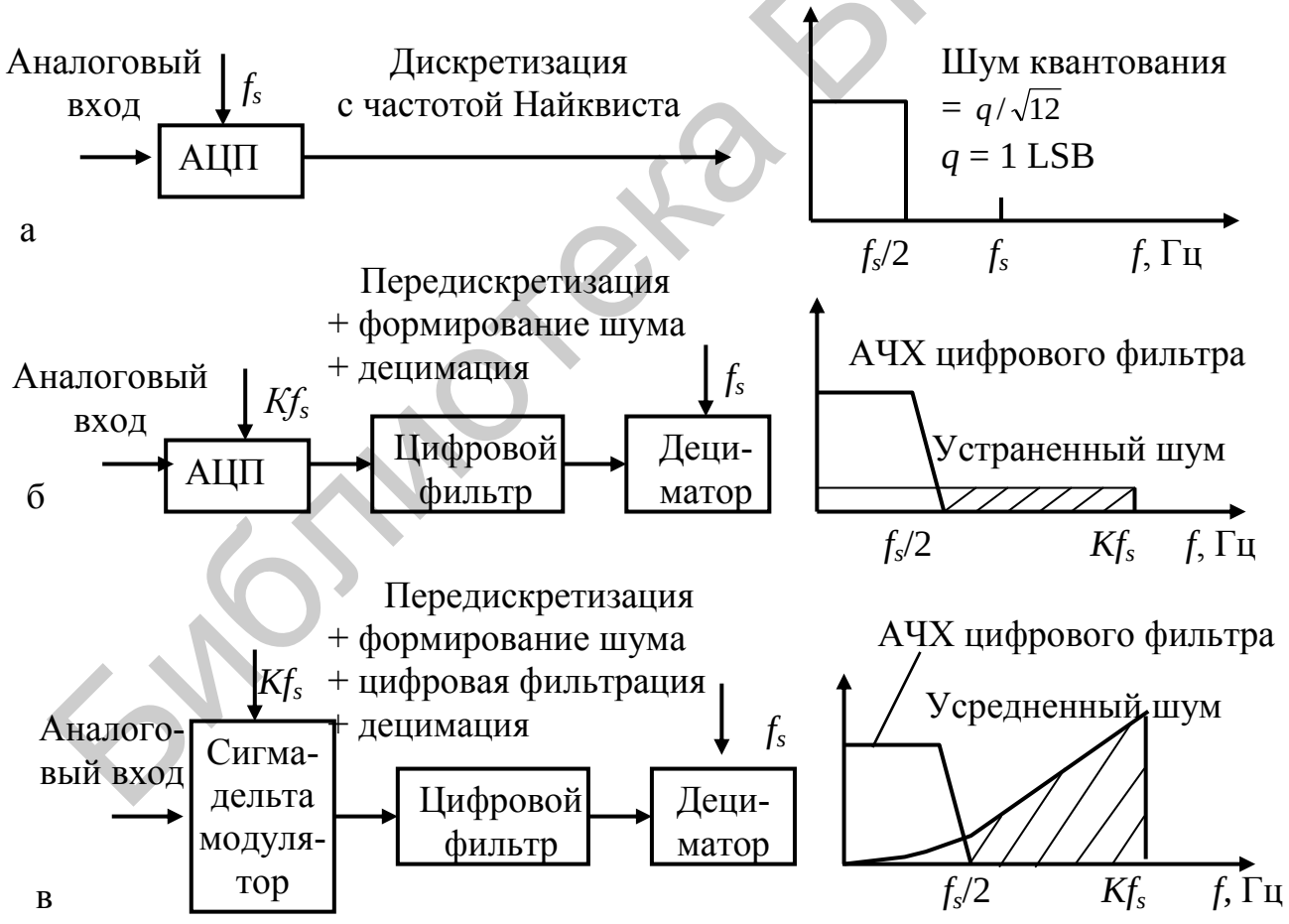


Рис. 13. Дискретизация с частотой Найквиста (а), с передискретизацией (б) и сигма-дельта модуляция (в)

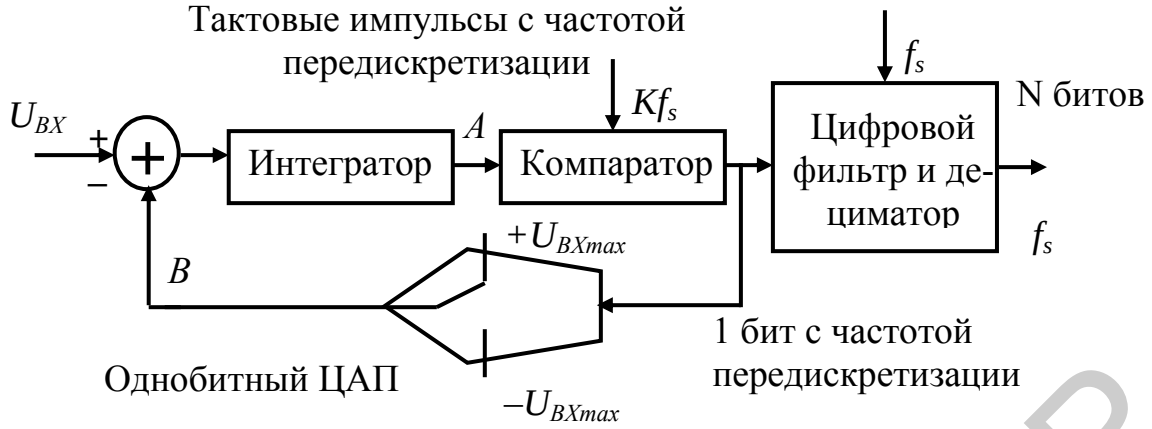


Рис. 14. АЦП с сигма-дельта модулятором первого порядка

Поэтому сигма-дельта модулятор можно рассматривать и как синхронный преобразователь напряжение-частота. Если число единиц в выходном битовом потоке рассчитывается на большом количестве отсчетов, выход счетчика представляется цифровым значением входного сигнала. Очевидно, этот метод усреднения будет работать только для постоянного входного напряжения или очень медленно изменяющихся сигналов. Кроме того, тактовые импульсы должны следовать с частотой 2^N, чтобы обеспечить эффективное разрешение в N битов, поэтому в этом случае снижается величина эффективной скорости дискретизации.

Нужно заметить, что поскольку цифровой фильтр представляет собой часть сигма-дельта АЦП, часто имеется встроенная задержка, называемая *pipeline* (иногда обозначается термином *latency*, означающим, что команды следуют непосредственно друг за другом).

Работу сигма-дельта модулятора легче понять, разобравшись в его отличиях от принципа работы ИКМ. В ИКМ входной сигнал подвергается скалярному квантованию на частоте Найквиста и осуществляется N-битное представление сигнала, т. е. обеспечивается квантование с 2^N уровнями. Кроме того, использование последовательных аппроксимирующих регистров и других устройств с высокой степенью разрешения весьма затруднительно в системах ИКМ, что обусловлено высокой точностью представления квантованных уровней и вследствие этого сложностью цепей, реализующих эти функции. Это заставляет разработчиков алгоритмов преобразования аналоговых сигналов в цифровую форму обращаться к методам сигма-дельта модуляции, использующейся в форме PDM, которая использует дискретизацию с перекрытием и достаточно сложные фильтрующие устройства, чтобы работать с низкобитными квантователями с высокой степенью разрешения.

Рассмотрим линейную модель дельта-сигма модулятора, которая особенно полезна в понимании устройств фильтрации и формировании шума, хотя, конечно, схема сигма-дельта модулятора является сугубо нелинейным устройством. Поэтому есть ряд явлений, которые невозможно объяснить с позиций линейной теории, например, такие как нестабильность или предельные циклы.

1.7. Линейная модель сигма-дельта модулятора и ИКМ

Рассмотрим дискретизацию с перекрытием, т. е. с частотой дискретизации более высокой, чем этого требует условие Найквиста. Скорость дискретизации в этом случае выбирается равной $f_s = 2^{r+1} \cdot f_B$, где f_B – верхняя частота спектра дискретизируемого сигнала, 2^r – отношение частоты дискретизации с перекрытием к частоте Найквиста:

$$OSR = 2^r = f_s/2f_B.$$

Таким образом, шум квантования расширяется до очень больших частот. Мощность шума квантования можно найти интегрированием спектральной плотности мощности по полосе частот, равной полосе частот сигнала

$$\sigma_n^2 = \int_{-f_B}^{f_B} S_e(f)df = \frac{2\sigma_n^2 f_B}{f_s} = \frac{\sigma_e^2}{OSR}, \qquad (15)$$

где $S_e(f) = \sigma_e^2 / f_B$ – спектральная плотность мощности шума в полосе частот сигнала.

Большая часть мощности шума квантования теперь находится вне полосы частот сигнала. Как следует из рис. 16, мощность шума квантования в полосе сигнала теперь уменьшается пропорционально коэффициенту OSR. Мощность сигнала в его полосе остается неизменной и дается соотношением (7). Теперь отношение сигнал/шум получается равным

$$SNR, дБ = 10\lg\frac{\sigma_x^2}{\sigma_e^2} + 10\lg 2^r \approx 20\lg A + 6{,}02b + 3{,}01r + 1{,}76. \qquad (16)$$

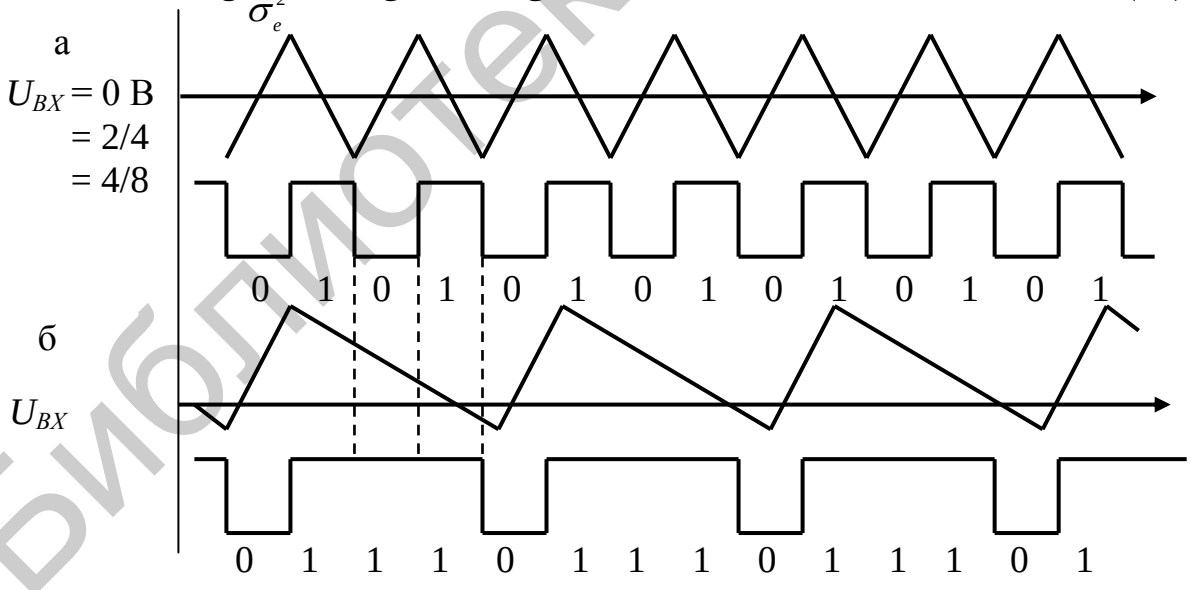


Рис. 15. Выходные напряжения сигма-дельта модулятора при $U_{BX} = 0$ (а) и $U_{BX} = +U_{BXmax}/2$ (б)

Таким образом, для каждого удвоения отношения частоты дискретизации с перекрытием отношение SNR увеличивается на 3 дБ. Дискретизация с перекрытием демонстрирует средство уменьшения количества битов в квантователе.

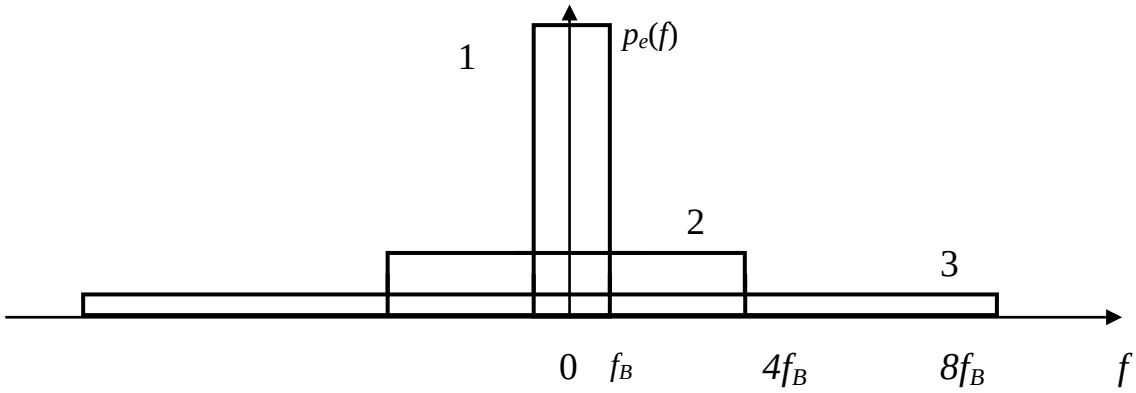


Рис. 16. Распределение шума квантования для скорости дискретизации

Пример 5. Оценить величину коэффициента OSR и частоту дискретизации, при которой можно достичь CD-качества записи (16-битный формат, частота дискретизации 44,1 кГц) с 8-битным квантователем.

Решение

Поскольку качество записи определяется отношением SNR, то это значит, что SNR при частоте дискретизации 44,1 кГц должно быть таким же, как при новой частоте дискретизации, но в 8-битном формате, т. е.

$$20lgA + 6{,}02\cdot 16 + 1{,}76 = 20\lg A + 6{,}02\cdot 8 + 3{,}01\cdot r + 1{,}76.$$

Отсюда получаем $r = 2^{16}$. Новая частота дискретизации в 2^{16} раз превышает частоту 44,1 кГц, т. е. равна $44{,}1\cdot 2^{16} = 2890137{,}6$ кГц. Реализовать это экспериментально пока невозможно.

1.8. Функции передачи сигнала и шума в линейной модели сигма-дельта модулятора

Представим сигма-дельта модулятор в терминах *Z*-преобразования в виде структурной схемы рис. 17. Для этого разместим перед квантователем фильтр с передаточной характеристикой *H(z)*. Имеем теперь систему, которую можно представить функциями, приложенными как к входному сигналу, так и к шуму. В *Z*-области выход SDM можно записать в виде

$$Y(z) = STF(z)\cdot X(z) + NTF(z)\cdot E(z), \qquad (17)$$

где STF – передаточная функция сигнала, NTF – передаточная функция шума. Чтобы найти их значения, заметим, что на вход фильтра воздействует сигнал $X(z) - E(z)$, так что

$$Y(z) = H(z)\cdot[X(z) - Y(z)] + E(z). \qquad (18)$$

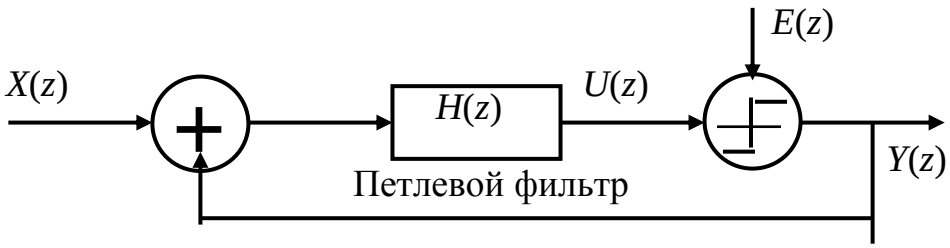


Рис. 17. Обобщенная структурная схема сигма-дельта модулятора с простой петлей обратной связи

Перепишем (18) иначе:

$$Y(z)[1 + H(z)] = H(z)\,X(z) + E(z). \qquad (19)$$

Таким образом,

$$STF(z) = H(z)/[1 + H(z)],\ NTF(z) = 1/[1 + H(z)]. \qquad (20)$$

Очевидно, что

$$STF(z) + NTF(z) = 1.$$

Полученные соотношения показывают, что выход сигма-дельта модулятора можно представить в виде суммы отфильтрованного входного сигнала и ошибки квантования. Следовательно, характеристики выполнения сигма-дельта модулятора можно рассматривать как определяемые входным сигналом, ошибкой квантования и петлевым фильтром.

Эти соображения подразумевают, что петлевой фильтр $H(z)$ должен иметь задержку по крайней мере на один тактовый интервал. Поэтому порядок числителя $H(z)$ должен быть по крайней мере на единицу меньше порядка знаменателя.

Линейная модель позволяет оценить влияние выбора фильтрующей функции на сигнал и шум. Схема фильтрования и цепь обратной связи вместе с дискретизацией с перекрытием являются существенными элементами SDM. Главная задача устройства SDM – выбрать фильтр с высокой стабильностью и некоторыми артефактами, так что в полосе прозрачности фильтра имеем

$$STF(z) = 1,\ NTF(z) = 0\ . \qquad (21)$$

Если этот случай будет реализован, то шум будет отфильтрован от полосы прозрачности и сигнал пройдет через фильтр без каких-либо изменений.

1.9. Линейная модель сигма-дельта модулятора

В большистве случаев входные сигналы и передаточные функции сигма-дельта модулятора определяются техническим заданием на разработку сигма-дельта модулятора, а что касается свойств ошибки квантования, то о них мало что известно. Кроме того, очень трудно выполнить строгий анализ ошибки квантования по той причине, что квантователь является сугубо нелинейной системой. Тем не менее при построении линейной модели сигма-дельта модулятора обычно делают три допущения относительно ошибки квантования,

позволяющие значительно упростить анализ работы сигма-дельта модулятора, считая, что последняя:

- равномерна распределена в области (-*q*/2, *q*/2);
- представляет собой стационарную последовательность типа белого шума;
- некоррелирована с последовательностью выходного сигнала или входом квантователя.

Первые два допущения характеризуют мощность и спектральную плотность ошибки квантования, третье допущение упрощает спектральный анализ модулятора за счет использования ортогональности входного сигнала и ошибки квантования. Если отказаться от третьего допущения, тогда не будет взаимной корреляции спектра входного сигнала и ошибки квантования внутри спектра выхода модулятора. Следовательно, спектр выходного сигнала можно выразить

$$S_{yy}(\omega) = \frac{|H(e^{i\omega})|^2}{1+|H(e^{i\omega})|^2} S_{xx}(\omega) + \frac{1}{|1+H(e^{i\omega})|^2} S_{ee}(\omega),$$

что соответствует допущениям линейной модели модулятора. Однако эти допущения справедливы не всегда, а только при определенных условиях.

W.R. Bennett первым указал (еще в 1948 году), что если следующие условия спрпаведливы, тогда принятые нами допущения будут хотя бы приблизительно корректны для систем импульсной кодовой модуляции:

- вход квнтователя не находится в области перегрузки по крутизне;
- уровень квантования является асимптотически большим;
- ширина бина 2*q* асимптотически мала;
- функция совместной плотности вероятности входного сигнала квантователя при отсчетах, взятых в различные моменты времени, является гладкой.

Конечно, эти условия в общем случае неверны, во-первых, потому, что обычно сигма-дельта модулятор использует однобитный квантователь с шириной бина, соответствующей уровню мощности входного сигнала в цепи обратной связи, и, во-вторых, дискретные значения выхода квантователя снова поступают на его вход. Следовательно, функция совместной плотности вероятности входного сигнала никогда не может быть гладкой.

Совсем неудивительно, что ошибка квантования не является белым шумом, поскольку она содержит определенные дискретные всплески, называемые свободными тонами или шумом модели. Удивительно, однако, что при моделировании и в реальных цепях модуляторов высокого порядка появляются ошибки квантования, такие, что они воспринимаются как белый шум. Но системы высокого порядка характеризуются нестабильными характеристиками, объяснить которые линейная модель сигма-дельта модулятора не может.

1.10. Анализ сигма-дельта модулятора первого порядка в Z-области

Рассмотрим схему сигма-дельта модулятора, приведенную на рис. 18, в *Z*-области. Квантователь можно описать как сложение сигнала $U(z)$, поступающего с выхода интегратора с шумом $E(z)$. Тогда

$$Y(z) = U(z) + E(z), \tag{22}$$

откуда

$$U(z) = Y(z) - E(z). \tag{23}$$

Исходя из рис. 22, запишем

$$U(z) = z^{-1} \cdot U(z) + X(z) - z^{-1} \cdot Y(z). \tag{24}$$

Подставив (24) в (23), получим

$$Y(z) - E(z) = z^{-1} \cdot U(z) + X(z) - z^{-1} \cdot [U(z) + E(z)] = \\ = X(z) - z^{-1} \cdot E(z), \tag{25}$$

т. е.

$$Y(z) - E(z) = X(z) - z^{-1} \cdot E(z). \tag{26}$$

Отсюда

$$Y(z) = X(z) - z^{-1} \cdot E(z) + E(z) = X(z) + (1 - z^{-1}) \cdot E(z). \tag{27}$$

Из полученного соотношения видим, что входной сигнал без искажений передается на выход. Квантование шума, к сожалению, проявляется как компонента в выходном сигнале и формируется передаточной функцией $H(z) = 1 - z^{-1}$. Эта передаточная функция соответствует фильтру верхних частот первого порядка. Результат квантования шума можно перенести в область очень высоких частот путем повышения частоты квантования и затем с помощью цифрового фильтра низких частот почти полностью подавить. Такой перенос шума в область высоких частот называется noise-shaping – формированием шума. Выполнение этой операции можно значительно усилить, использовав в качестве передаточной функции шума фильтры высоких частот высокого порядка. Это реализуется в сигма-дельта модуляторах высших порядков.

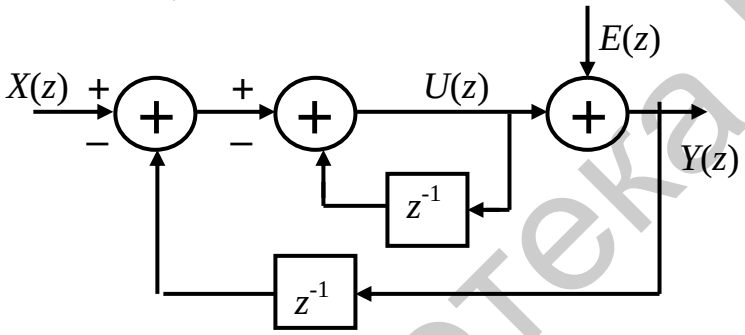


Рис. 18. Сигма-дельта модулятор в Z-области

1.11. Анализ сигма-дельта модулятора второго порядка в Z-области

Сигма-дельта модулятор второго порядка получается в результате расширения петли обратной связи за счет введения новых интеграторов. На рис. 19 приведена структурная схема такого модулятора второго порядка.

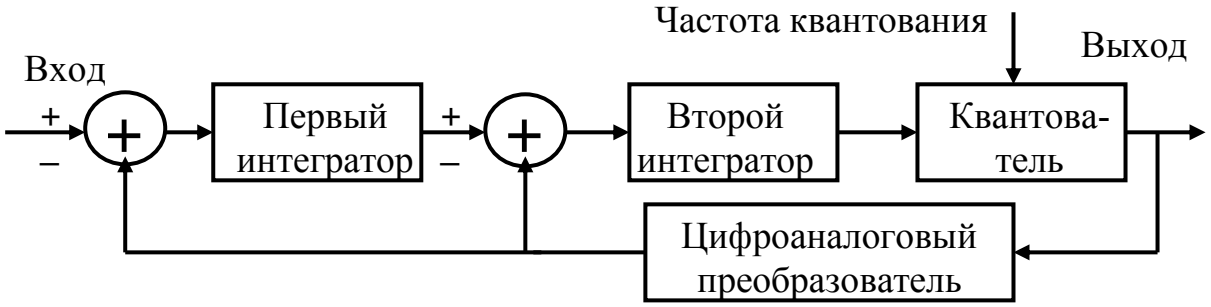


Рис. 19. Структурная схема сигма-дельта модулятора второго порядка

Для исследования влияния расширения петли обратной связи на характеристики такого модулятора представим структурную схему его в *Z*-области, как показано на рис. 20.

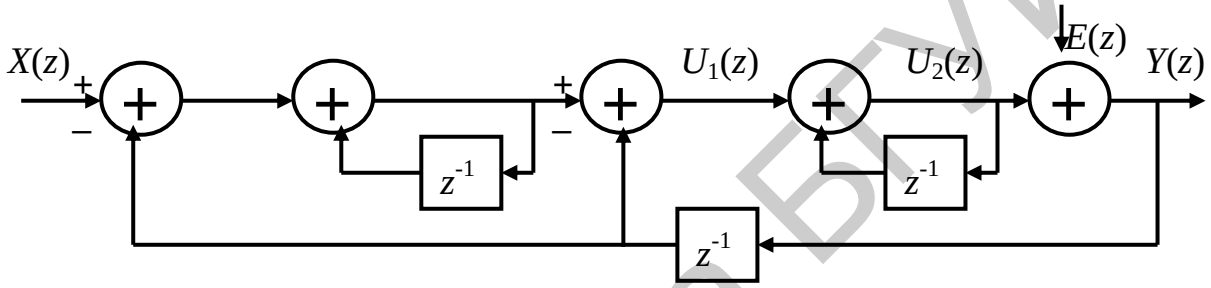


Рис. 20. Сигма-дельта модулятор второго порядка в Z-области

Опишем работу схемы с помощью уравнений

$$Y(z) = U_2(z) + E(z),$$
$$U_2(z) = Y(z) - E(z),$$
$$U_1(z) = z^{-1}\cdot U_1(z) + X(z) - z^{-1}\cdot Y(z),$$
$$U_1(z) - z^{-1}\cdot U_1(z) = X(z) - z^{-1}\cdot Y(z),$$
$$U_1(z) = [X(z) - z^{-1}\cdot Y(z)]/(1 - z^{-1}),$$
$$U_2(z) = z^{-1}\cdot U_2(z) + U_1(z) - z^{-1}\cdot Y(z).$$

Отсюда легко получить

$$Y(z) - E(z) = z^{-1}\cdot U_2(z) + U_1(z) - z^{-1}\cdot Y(z) =$$
$$= z^{-1}\cdot U_2(z) + U_1(z) - z^{-1}\cdot [\, U_2(z) + E(z)] =$$
$$= U_1(z) - z^{-1}\cdot E(z).$$

И окончательно получаем

$$Y(z) - E(z) = [X(z) - z^{-1}\cdot Y(z)]/(1 - z^{-1}) - z^{-1}\cdot E(z),$$
$$(1 - z^{-1})[\, Y(z) - E(z)] = \quad X(z) - z^{-1}\cdot Y(z) - z^{-1}\cdot(1 - z^{-1})\cdot E(z),$$
$$Y(z) = X(z) + (1 - z^{-1})^2\cdot E(z).$$

Полученное соотношение показывает, что сигма-дельта модулятор второго порядка также переносит входной сигнал без искажений на выход с добавлением шума квантования. Конечно, передаточная функция теперь стала другой, а именно $H(z) = (1 - z^{-1})^2$. Она соответствует фильтру верхних частот второго порядка. Шумы в области низких частот также сильнее подавляются, чем у сигма-дельта модулятора первого порядка. Рис. 21 дает

наглядное представление о частотных зависимостях уровня шума квантования в сигма-дельта модуляторах первого и второго порядков.

1.12. Сигма-дельта модулятор с полосовым фильтром

Если заменить интеграторы в сигма-дельта АЦП полосовыми фильтрами, как это показано на рис. 26, тогда шум квантования будет перемещаться вверх и вниз по частоте, оставляя свободную зону в полосе пропускания. Если цифровой фильтр запрограммировать так, что он будет иметь полосу прозрачности, то получим сигма-дельта АЦП с полосой пропускания, что будет предпочтительнее ФНЧ. Такие устройства будут полезными при непосредственном преобразовании сигнала промежуточной частоты в цифровой сигнал во многих конкретных приложениях. Однако модулятор и цифровой фильтр должны быть разработаны для специфического множества частот, требующихся в этих системах. Обсудим некоторые ограничения применительно к этим приложениям.

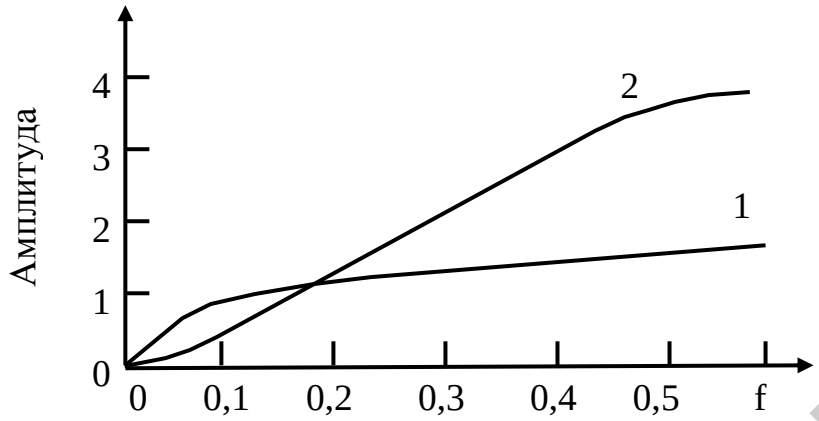


Рис. 21. Передаточные функции шума в сигма-дельта модуляторах первого (1) и второго (2) порядков

При обычной дискретизации полоса пропускания сигма-дельта АЦП должна быть по крайней мере в два раза превышать полосу реальных частот сигнала. Центральная частота полосового фильтра должна соответствовать несущей частоте f_s обрабатываемого сигнала. Если речь идет о промежуточной частоте 465 кГц и полосе частот сигнала в 10 кГц, то коэффициент передискретизации должен удовлетворять условию Kf_s = 20 МГц и скорость дискретизации должна равняться f_s = 20 кГц, обеспечивая при этом динамический диапазон в полосе частот сигнала порядка 70 дБ. В литературе описаны и другие сигма-дельта АЦП, например, AD9870 для номинальной частоты передискретизации 18 МГц на центральной частоте 2,25 МГц и полосой от 10 до 150 кГц.

1.13. АЦП с сигма-дельта модулятором первого порядка

Сигма-дельта модулятор первого порядка имеет простой интегратор в цепи обратной связи с фильтром. Простейшее устройство не имеет никаких усилительных элементов и во временной области (рис. 22, а) может быть представлено следующим выражением:

$$u(n+1) = x(n) - y(n) + u(n). \qquad (28)$$

Обозначая $e_q = Q - u$, и учитывая предыдущий временной шаг, получаем

$$y(n) = z(n-1) + e_q(n) - e_q(n-1). \quad (29)$$

Структурная схма Z-преобразования приведена на рис. 22,б. Соответствующее уравнение в терминах Z-преобразования имеет вид

$$Y(z) = X(z)z^{-1} + E(z)(1 - z^{-1}) \quad (30)$$

Передаточная функция сигнала равна z^{-1}, преобразование сигнала сводится к его задержке на один такт. Передаточная функция шума равна $1 - z^{-1}$ и сдвигает шум в сторону высоких частот. Используя тригонометрические преобразования, можно показать, что

$$|NTF(f)| = (1-e^{-i2\pi f/f_s})(1-e^{-i2\pi f/f_s}) = 4\sin^2(\pi f/fs). \quad (31)$$

В отличие от ИКМ с дискретизацией с перекрытием, которая имеет равномерную NTF, формирование шума в SDM предполагает непостоянную мощность шума, в полосе частот сигнала определяющуюся следующим образом:

$$\sigma_x^2 = \int S_e(f)|\, NTF(f)\,|^2\, df\,, \quad (32)$$

где $S_e = \sigma_x^2 / f_s$ – спектральная плотность мощности шума квантования, не подвергавшегося формированию. Общая мощность шума σ_e^2 остается неизменной, но теперь шум сдвигается в область высоких частот. Это показано на рис. 23, который сравнивает спектральную плотность мощности SDM первого порядка со спектральной плотностью мощности ИКМ.

Осуществляя дискретизацию с большим коэффициентом OSR, т. е. выбирая $f_s >> f_B$, и используя ряд Тейлора для функции $\sin x = x - x^3/3! + x^5/5! \ldots$, уравнение (26) можно решить, полагая

$$\sigma_n^2 = \frac{\sigma_e^2}{f_s}\left[4f_B - \frac{2f_S}{\pi}\sin(2\pi f_B / f_s)\right] = \frac{\pi^2}{3\cdot OSR^3}\sigma_e^2. \quad (33)$$

Отношение сигнал/шум теперь можно записать в виде

$$SNR, дБ = 10\lg\frac{3\cdot 2^{3r}\cdot\sigma_x^2}{\pi^2\cdot\sigma_e^2} \approx 20\lg A + 6{,}02b + 9{,}03r - 3{,}41. \quad (34)$$

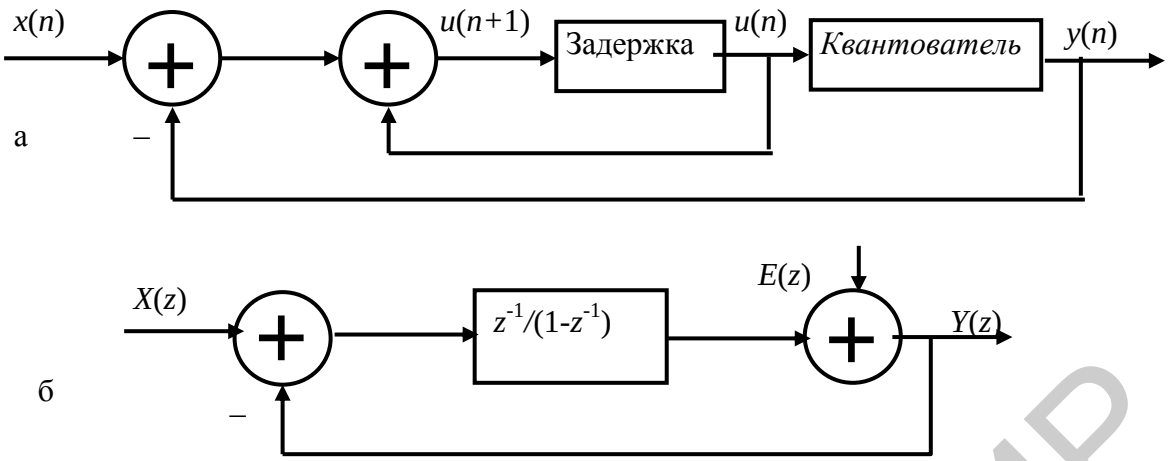


Рис. 22. Структурные схемы SDM модулятора первого порядка (а) и Z-преобразования с квантователем, аппроксимированным источником шума (б)

Рис. 23. Нормированные спектральные плотности для PCM и для SDM первого порядка

Результат использования формирования шума устройством первого порядка очевиден. Теперь получается улучшение на 9 дБ на каждое удвоение коэффициента OSR, что значительно больше улучшения в 3 дБ, что имело место без формирования шума.

1.14. SNR для сигма-дельта модулятора высоких порядков

Анализ, проведенный выше, легко можно распространить и на SDM более высоких порядков. Передаточная функция SDM *N*-го порядка имеет вид

$$Y(z) = X(z)z^{-1} + E(z)(1 - z^{-1})^N = STF(z)X(z) + NTF(z)E(z). \quad (35)$$

Таким образом, передаточная функция шума имеет вид

$$NTF(f) = (1 - e^{-i2\pi f/f_s})^N. \quad (36)$$

Используя формулу (29), получаем

$$\sigma_n^2 = \frac{\sigma_e^2}{f_s}\int [2\sin(\pi f / f_s)]^{2N}\, df \approx \frac{\pi^{2N}\cdot\sigma_e^2}{(2N+1)OSR^{2N+1}}, \qquad (37)$$

где приближенное равенство было получено использованием ряда Тейлора, в котором было удержано первое ненулевое слагаемое. Сравнение с SDM первого порядка указывает на большее подавление шума квантования в области низких частот и на большее усиление шума за пределами полосы частот сигнала. Соотношение (31) можно использовать, чтобы найти общую формулу для SNR идеального SDM *N*-го порядка:

$$SNR,\ дБ = 10\lg\frac{\sigma_x^2}{\sigma_e^2} + 10\lg\frac{(2N+1)2^{(2N+1)r}}{\pi^{2N}} \approx$$

$$\approx 20\lg A + 6{,}02b + 1{,}76 + 10\lg(2N+1) - 9{,}94N + 3{,}01(2N+1)r. \qquad (38)$$

Очевидно значительное улучшение отношения сигнал/шум с увеличением порядка SDM. Например, для SDM второго порядка (N = 2) имеет место улучшение SNR на 15 дБ с каждым удвоением величины OSR.

Вообще для SDM *N*-го порядка имеет место улучшение в отношении сигнал/шум, составляющее 3(2N + 1) дБ на каждое удвоение коэффициента OSR и на 6 дБ на каждый дополнительный бит в квантователе. Таким образом, использование SDM высоких порядков обеспечивает значительно лучшее отношение сигнал/шум, чем получение этой же величины отношения за счет простого увеличения количества битов квантователя.

Конечно, это аппроксимация. Она зависит от коэффициентов модулятора, от аппроксимаций, использованных при выводе соотношений, и многих других факторов. Во всяком случае этот подход обеспечивает верхний предел SNR.

1.15. MASH-архитектура сигма-дельта модуляторов

Сигма-дельта модуляторы первого порядка не очень подходят для применений, требующих высокого разрешения. Кроме того, шум квантования часто рассматривается как высококоррелированный с входным сигналом. Между тем разработка, например, сигма-дельта модулятора третьего порядка из-за появления трех полюсов в его NTF создает проблему стабильности работы устройства в широкой полосе рабочих частот. С целью снижения взаимного влияния противоречивых тенденций (снижение уровня шума квантования в полосе частот обрабатываемого сигнала и обеспечение высокой стабильности) предложена техника производства сигма-дельта модуляторов высокого порядка с гарантированной стабильностью работы, называемая MASH-архитектурой (Multi stAge noise SHape, MASH), в которой используется каскадное соединение нескольких первого или второго порядка сигма-дельта модуляторов. Рассмотрим работу такого модулятора, структурная схема которого показана на рис. 24.

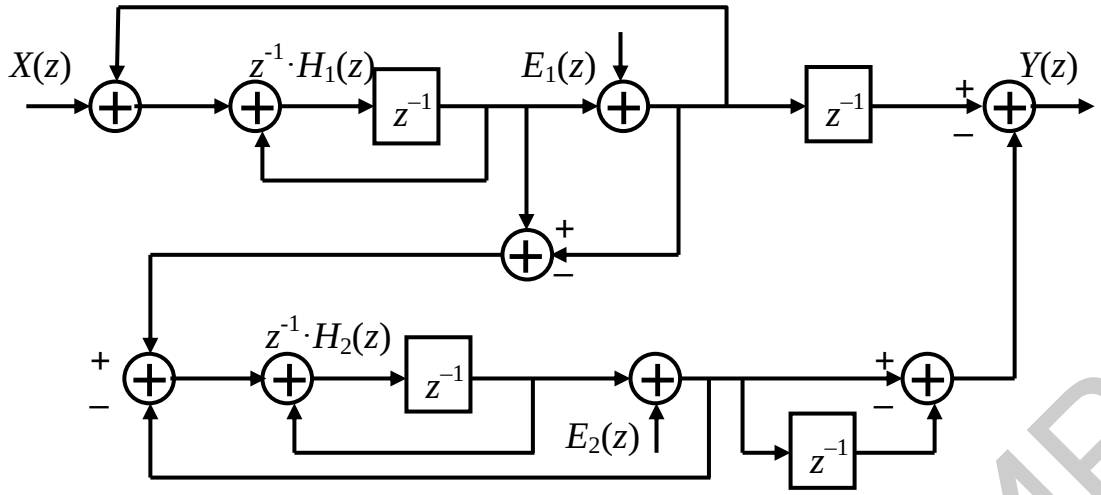


Рис. 24. Каскадное соединение двух сигма-дельта модуляторов (MASH-архитектура)

Преимущества MASH-архитектуры в стабильности работы и свободе модели от шума на выходе модулятора, если каскадно соединенные модели достаточно большие и хорошо продумана их конструкция. Сигма-дельта модуляторы MASH-архитектуры требуют высокой точности реализации коэффициентов усиления и согласования сопротивлений между соседними составляющими архитектуры.

Проведем описание в *Z*-области приведенного на рис. 24 сигма-дельта модулятора. Имеем

$$Y(z) = \frac{z^{-1}H_1(z)X(z)}{1+z^{-1}H_1(z)} + \frac{z^{-1}E_1(z)}{1+z^{-1}H_1(z)} - \frac{z^{-1}E_1(z)H_1(z)}{[1+z^{-1}H_2(z)]H_3(z)} - \frac{E_2(z)}{[1+z^{-1}H_2(z)]H_3(z)}.$$

Если выполнено условие $H_1(z) = H_2(z)$, то последнее выражение можно упростить и представить в виде

$$Y(z) = \frac{z^{-2}H_1(z)X(z)}{1+z^{-1}H_1(z)} + \frac{z^{-1}E_1(z)[H_3(z)-H_1(z)]}{1+z^{-1}H_1(z)} - \frac{E_2(z)}{[1+z^{-1}H_1(z)]H_3(z)}.$$

Если положить, что $H_1(z) = H_3(z)$, то тогда получим

$$Y(z) = \frac{z^{-2}X(z)H_1(z)}{1+z^{-1}H_1(z)} - \frac{E_2(z)}{[1+z^{-1}H_1(z)]H_1(z)}.$$

Если в качестве $H_1(z)$ использовать АЧХ ФНЧ, т. е. положить $H_1(z) = 1/(1 - z^{-1})$, тогда получим

$$Y(z) = z^{-2}X(z) - (1-z^{-1})^2E_2(z).$$

1.17. Предельные циклы

Рассмотрим однобитный SDM первого порядка, который описывается уравнением (16) с постоянным напряжением на входе $x = 0{,}5$ и начальными условиями, $u(n) = 0{,}1 > 0$. Имеем

$$u(n+1) = 0{,}5 - 1 + 0{,}1 = = -0{,}4,$$
$$u(n+2) = 0{,}5 + 1 - 0{,}4 = -1{,}1,$$
$$u(n+3) = 0{,}5 - 1 + 1{,}1 = 0{,}6,$$

$$u(n+4) = 0{,}5 - 1 + 0{,}6 = 0{,}1. \quad (39)$$

Таким образом, входное напряжение, поступающее на квантователь, повторяется с периодом в четыре такта, и квантователь производит повторяющийся выходной поток +1, –1, +1, +1.

Заметим, что среднее значение выходного напряжения за четыре итерации составляет [3*(+1)+1*(–1)]/4=0,5, т. е. является тем же самым, что и на входе.

Для SDM первого порядка, пока выполняется уравнение (16), получаем

$$u(n+k) = \sum_{i=0}^{k-1} x(n+1) - \sum_{i=0}^{k-1} y(n+i) + u(n) \quad . \quad (40)$$

Предполагаем, что среднее значение выходного сигнала равно входному сигналу, получаем после *k* циклов.

Наличие повторяющейся последовательности в выходном битовом потоке называется предельным циклом. Наличие предельного цикла создает проблемы в обработке сигналов с помощью SDM. В только что рассмотренном примере предельный цикл проявляется в существовании последовательности прямоугольных импульсов со скважностью 4/3.

2. Порядок выполнения работы

Задание 1. Приведенная ниже программа показывает, как функция quantiz использует параметры partition и codebook для изображения перехода реального вектора samp к новому вектору quantized, который задан точками –1, 0.5, 2, 3.

```
>>partition=[0,1,3];
>>codebook=[-1, 0.5,2,3];
>>samp=[-2.4,-1,-0.2,0,.2,1,1.2,1.9,2,2.9,3,3.5,5];
>>[index,quantized]=quantiz(samp,partition,codebook);
>>quantized
```

Запустите эту программу на выполнение и прокомментируйте полученные результаты.

Замечание. Параметр квантования partition определяет несколько неперекрывающихся областей значений внутри множества действительных чисел. Чтобы охарактеризовать partition как параметр, нужно перечислить конечные точки разных областей, описываемых вектором. Например, если partition разделяет действительные числа на множества {x:x≤0}, {x:0<x≤1}, {x:1<x≤3}, {x:x>3}, тогда можно представить partition в виде трехэлементного вектора [0,1,3]. Длина вектора partition на единицу меньше числа его интервалов. Codebook относится к квантователю, который назначает значения, что приходятся на каждую область partition. Например, вектор [–1,.5,2,3] является одним из возможных codebook для partition [0,1,3].

Задание 2. Приведенная ниже программа после квантования дискретизированного синусоидального сигнала дает графическое изображение исходного и квантованного сигналов.

```
>>t=[0:.1:2*pi];
>>sig=sin(t);
>>partition=[-1:.1:1];
>>codebook=[-1.2:.2:1];
```

```
>>[index,codebook]=quantiz(sig,partition,codebook);
>>plot(t,sig,'x',t,quant,'.')
>>legend('Исходный сигнал','Квантованный сигнал');
>>axis([-.2 7 -1.2 1.2])
```

Запустите эту программу на выполнение и прокомментируйте полученные результаты. Выполните эти же операции для косинусоидального сигнала с расстоянием между пиками, равным 6 В, и уровнями квантования, разделенными 0,3 В. Изобразите исходный и квантованный сигналы на одном и том же графике для двух периодов исходного сигнала. Изобразите на двух разных графиках эффект изменения codebook при включении входных пиковых значений исходного сигнала в значения квантованного сигнала.

Задание 3. Приведенная на рис. 25 модель, построенная в SIMULINK, показывает, как блок Qantizing Encoder использует параметры partition и codebook для представления действительного вектора в виде нового вектора, элементы которого имеют вид: –1,.5,2,3. В окне Scope верхний сигнал показывает исходный сигнал, нижний сигнал соответствует квантованному сигналу.

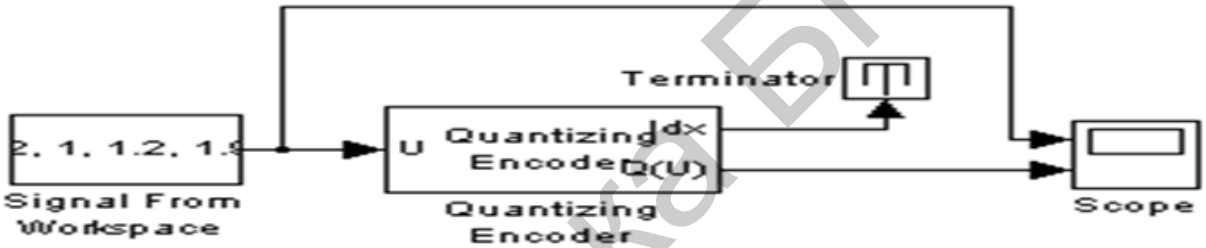


Рис. 25. Модель квантования, реализованная в SIMULINK

Выберите:

- Signal From Workspace из the Signal Processing Sources library и установите сигнал [–2.4,–1,–.2,0,.2,1,1.2,1.9,2.2,2.9,3.3,3.5]';

- В Quantizing Encoder используйте Quantization partition=[0,1,3] и Quantization codebook= [–1,.5,2,3];

- Terminator и Scope из Simulink Sinks library.

Из модели окна SIMULINK выберите Configurations parameters, в котором установите Stop time = 12. Запустите модель на выполнние и получите изображение сигналов на экране Scope, предварительно установив число осей, равное двум. Прокомментируйте полученные результаты моделирования работы скалярного квантователя.

Задание 4. Используя приведенную ниже программу, изучите работу дельта-модулятора, наблюдая создаваемый им шум при изменении амплитуды входного сигнала. Изобразите результаты работы дельта-модулятора при различных амплитудах входного сигнала, наблюдая гранулированный шум и шум,

возникающий в результате перегрузки модулятора по крутизне. Объясните получающиеся результаты.

```
>> close all;
>>clf;
>>t=0:0.1:500;%sample array
>>omega=pi/100;% frequency
>>A=5;% Amplitude
>>x=A*cos(omega*t);% Input signal
>>delta=.008;
>>s=zeros(1,length(x)+1);
>>xhat=zeros(1,length(x)+1);
>>xhat(1)=0;
>>s(1)=0;
% modulation
>>for i=2:length(x)
>>xhat(i)=xhat(i-1)+s(i-1);% output from predictor
>>d=x(i)-xhat(i);% error
>>if d>=0
>>s(i)=delta;
>>else
>>s(i)=-delta;
>>end
>>end
>>y=zeros(1,length(x));
>>y(1)=0;% demodulation
>>for i=2:length(x)-1
>>y(i)=y(i-1)+s(i-1);
>>end
% Plotting
>>subplot(2,1,1)
>>plot(t,x)
>>title('Input');
>>ylabel('Amplitude');
>>subplot(2,1,2)
>>plot(t,y)
>>title('Output');
>>xlabel('time [s]');
>>ylabel('Amplitude');
```

Задание 5. Замените входной синусоидальный сигнал, использовавшийся при выполнении задания 1, на сигнал треугольной формы и повторите операции по исследованию дельта-модулятора в предыдущем исследовании.

Задание 6. Замените входной синусоидальный сигнал, использовавшийся при выполнении задания 1, на сигнал пилообразной формы и повторите операции по исследованию дельта-модулятора в предыдущем исследовании.

Задание 7. Выполните предыдущие задания, моделируя поведение дельта-модулятора в системе SIMULINK. Обратите внимание на формирование гранулированного шума и шума вследствие перегрузки дельта-модулятора по крутизне. Для начала исследований можно воспользоваться приведенной схемой на рис. 26. Представьте в отчете наиболее важные результаты для всех генераторов блока «Signal Generator».

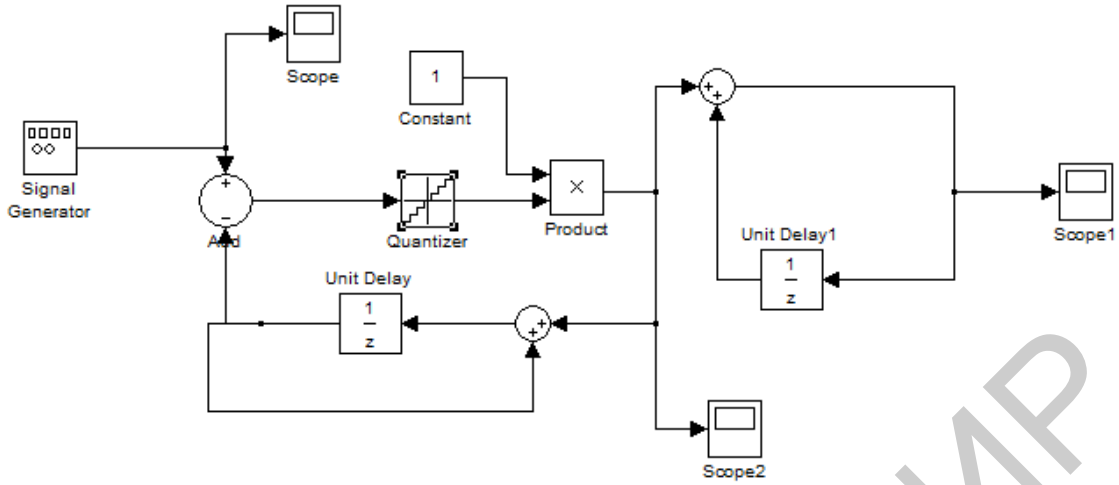


Рис. 26. SIMULINK-модель дельта-модулятора

Задание 8. Проведите исследование работы дельта-модулятора по модели, приведенной на рис. 27, на котором показана процедура преобразования аналогового сигнала в трехбитную цифровую форму. Прокомментируйте полученные результаты.

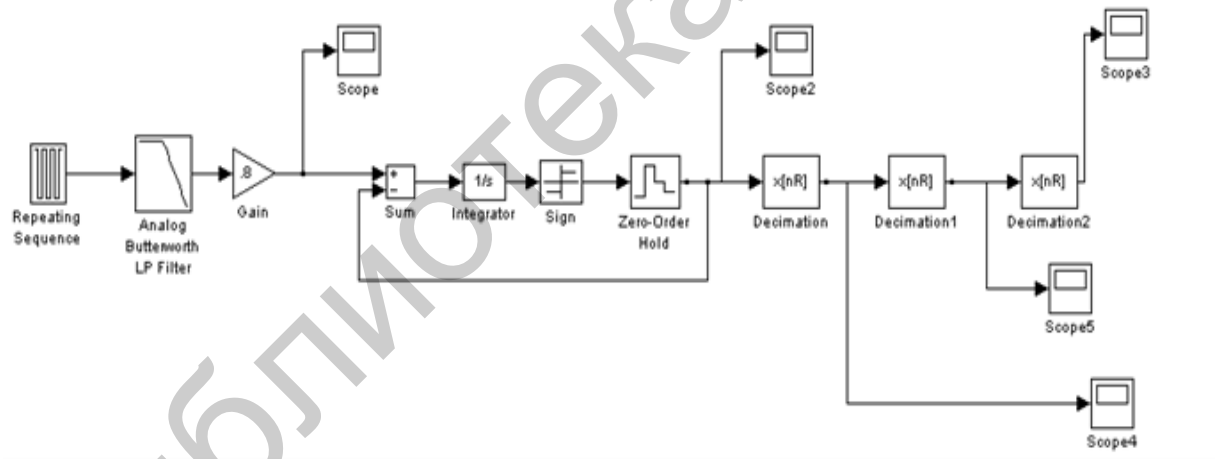


Рис. 27. Расширенная SIMULINK-модель дельта-модулятора

Задание 9. Проведите исследование работы сигма-дельта модулятора по модели, приведенной на рис. 28.

Можно воспользоваться схемой, показанной на рис. 27, и получить приведенные на рис. 29 результаты, получение которых она обеспечивает. Дополните схему, показанную на рис. 27, чтобы получить восстановленный сигнал.

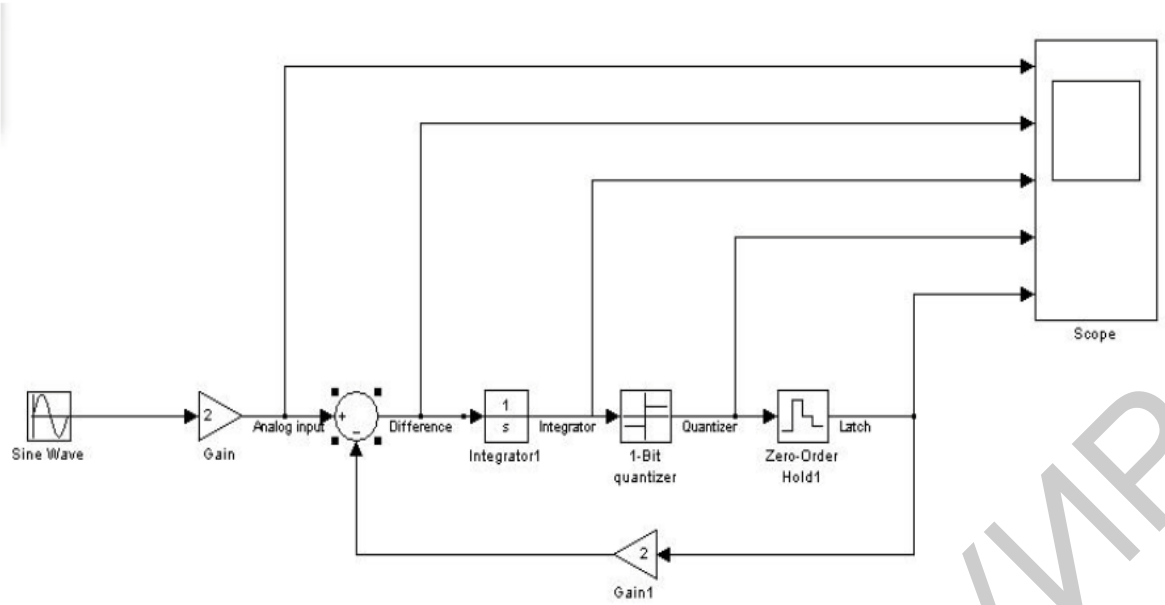


Рис. 28. SIMULINK-модель сигма-дельта модулятора

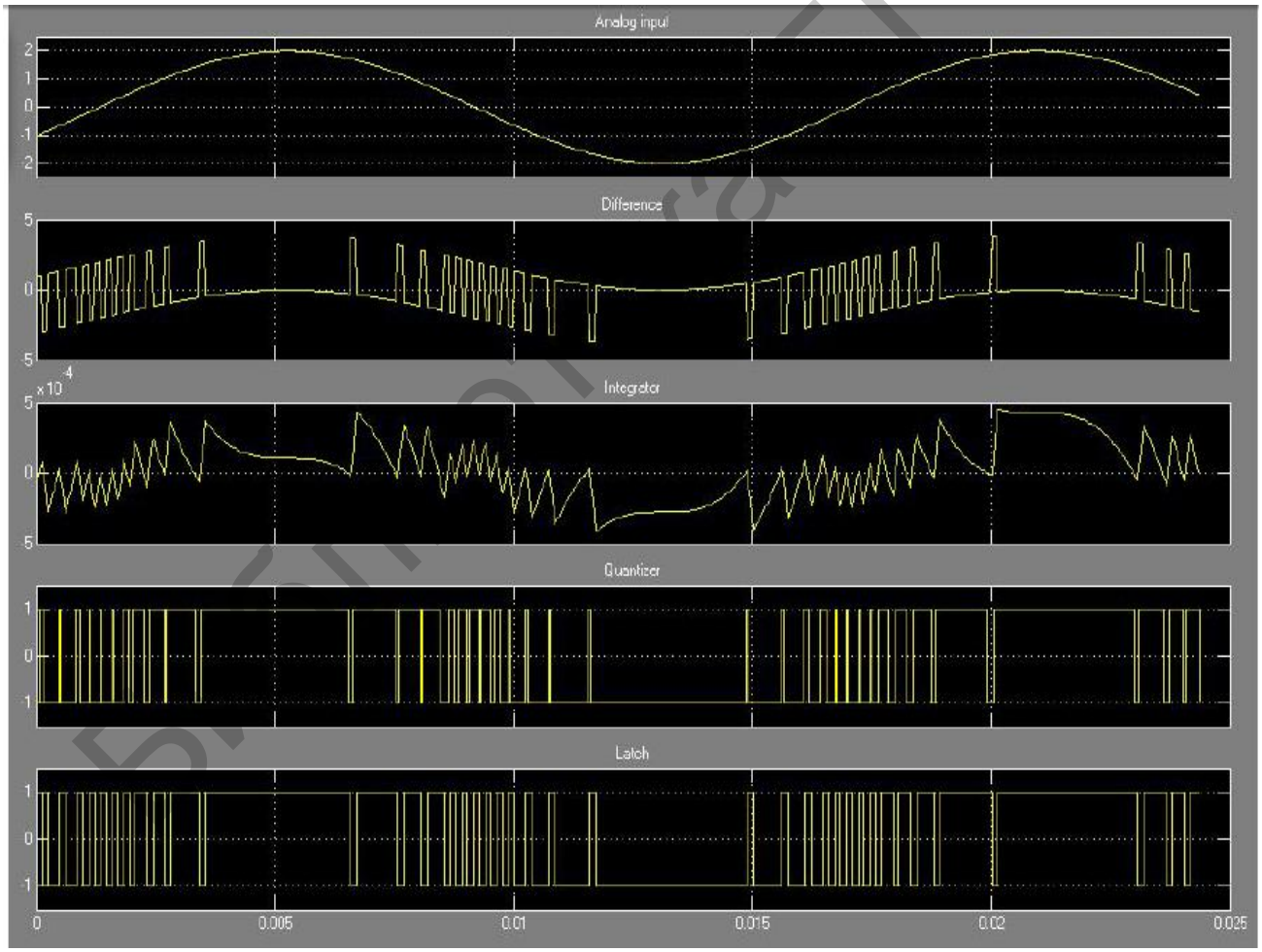


Рис. 29. Результаты моделирования работы сигма-дельта модулятора в SIMULINK-модели сигма-дельта модулятора, приведенной на рис. 35

Преимущество DPCM состоит в снижении количества информации, что будет передано по каналу связи при сохранении величины SNR или увеличение отношения SNR при сохранении объема передаваемой по каналу информации. Улучшение выполнения системы передачи информации, что можно получить в результате применения в ней DPCM, по сравнению с использованием PCM может составлять увеличение отношения SNR до величины порядка 20 дБ для некоторых сигналов. Это соответствует увеличению интенсивности сигнала по сравнению с сигналом шума в 100 раз или увеличению количества информации более чем 3 бита/отсчет. Однако есть по крайней мере две проблемы, которые при этом нужно учитывать:

- оценка накопления шума квантования;
- снижение влияния ошибок передачи информации.

Первую проблему можно решить путем устранения разностного сигнала квантования $d[k]$ между сигналом $x[k]$ и его предыдущим отсчетом $x[k-1]$ или

$$d[k]=x[k]-x[k-1]$$

и квантования вместо разностного сигнала, назовем его $g[k]$, что представляет собой разность между $x[k]$ и предыдущим отсчетом его квантованного значения $x_q[k-1]$. Поэтому $g[k[$ можно представить следующим образом:

$$g[k]=x[k]-x_q[k-1].$$

По-видимому, это потребует применения квантователя к сигналу $x[k]$, чтобы получить $x_q[k-1]$, хотя следует попытаться избежать выполнения этой операции, так как амплитуда $x[k]$ в общем случае превышает амплитуду разностного сигнала $d[k]$ или даже $g[k]$. Действительно, располагая сигналами $x[k]$ и $g[k]$, можно восстановить квантованную форму сигнала $x[k]$, используя следующую систему, показанную на рис. 30. С помощью этой системы просто доказать, что результирующий сигнал $x_q[k]$ представляет собой квантованное значение сигнала x[k]. Видим, что

$$g[k[=x[k]-x_q[k-1].$$

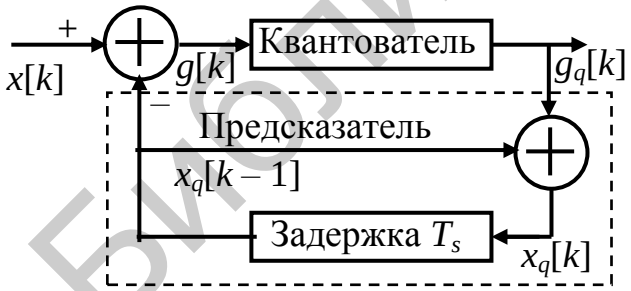


Рис. 30. Система восстановления квантованной формы $x[k]$

Теперь выходной сигнал квантователя – это квантованное значение $g[k]$, которое можно представить, добавив шум квантования $q[k]$ к входному сигналу квантователя. Поэтому

$$g_q[k]=g[k]+q[k]\ .$$

Подставив $g[k]$ и $g_q[k]$, получим

$$g_q[k]=x[k]-x_q[k-1]+q[k].$$

Из рис. 30 видно, что

$$x_q[k]=g_q[k-1]+x_q[k-1]=x[k]-x_q[k-1]+q[k]+x_q[k-1]=x[k]+q[k].$$

Поэтому действительно функция $x_q[k]$ – это квантованное значение $x[k]$, как это видно из окончания последней строки записанного только что уравнения, что совсем не значит, что если квантуется $x[k]$ непосредственно квантователем, то получим $x_q[k]$. Этот анализ только свидетельствует о том, что $x_q[k]$ –

квантованное значение $x[k]$. Если пропустить $x[k]$ через тот же самый квантователь в структурной схеме рис. 30, получим другую функцию $x_{q2}[k]$ с отчетами, что в общем случае отличается от $x_q[k]$.

На приемной стороне системы с DPCM можно использовать блок предсказателя с его входом, подключенным к выходу квантователя для получения сигнала $x_q[k]$. Поэтому структурную схему системы, приведенную на рис. 30, можно упростить и представить в виде, показанном на рис. 31.

Рассмотрим конкретную ситуацию, предположив, что квантователь, используемый для квантования сигнала $g[k]$, представляет собой 8-уровневый квантователь с интервалами квантования [–4, –3), [–3, –2), …, [3, 4) и выходными уровнями квантования, совпадающими с точками в каждом интервале квантования (–3,5, –2,5, …, 3,5). Результаты расчетов, выполненных с применением пакета MATLAB, приведены в табл. 2, которая показывает, что DPCM не вызывает накопления ошибки. Рассматривая восстановленный сигнал и исходный входной сигнал, видим, что амплитуда разности всегда меньше или равна половине интервала квантования, т. е. равна 0,5. даже когда ошибка квантования для последовательности отсчетов имеет то же самое направление, что имеет место в последних четырех столбцах таблицы, разность между входным и выходным сигналами системы всегда находится внутри половины интервала квантования.

Рис. 31. Упрощенная система восстановления квантованной формы $x[k]$

Что касается снижения влияния ошибки передачи, которая проявляется во всех восстановленных отсчетах входного сигнала, то лучший метод устранения этой проблемы – разделить данные на множество отсчетов и забыть о передатчике и приемнике после передачи каждого множества отсчетов. Таким образом, ошибки передачи, что имеют место, будут воздействовать только на отсчеты этой части данных, как только система сбрасывается, влияние ошибки устраняется.

Теперь приведем задание на изучение DPCM в системе SIMULINK на примере передачи гармонического сигнала по каналу с DPCM. Для этого рекомендуется собрать схему, приведенную на рис. 32, с помощью которой можно наблюдать искажения выходного сигнала в области перегрузки по крутизне. Объясните наблюдаемые результаты исследования схемы DPCM. Осуществляя передачу по каналу с шумом AWGN, обоснуйте выбор указанных на рис. 33…36 параметров блоков и запишите результаты экспериментальных исследований и объясните наблюдаемые искажения восстановленного сигнала.

Сделайте свои выводы относительно сравнения дифференциальной импульсно-кодовой амплитудной модуляции с параметрами дельта-модуляции.

Таблица 2

Расчеты, показывающие, что DPCM не вызывает накопления ошибки

k	–1	0	1	2	3	4	5	6	7	8	9
$x[k]$	0	0.3	1.5	0.7	1.0	2.3	3.7	2.8	3.4	2.8	0
$g[k]$ $(= x[k] - x_q[k-1])$	0	–0.2	1.5	–0.8	0	0.8	1.7	–0.7	0.4	–0.7	–3.0
$g_q[k]$	0.5	–0.5	1.5	–0.5	0.5	0.5	1.5	–0.5	0.5	–0.5	–2.5
Quantization Up/Down	U (or D)	D	–	U	U (or D)	D	D	U	U	U	U
$x_q[k]$ $(= g_q[k] + x_q[k-1])$	0.5	0	1.5	1.0	1.5	2.0	3.5	3.0	3.5	3.0	0.5
$x_q[k-1]$	0	0.5	0	1.5	1.0	1.5	2.0	3.5	3.0	3.5	3.0
$g_q[k]$	0.5	–0.5	1.5	–0.5	0.5	0.5	1.5	–0.5	0.5	–0.5	–2.5
$\hat{x}[k-1]$	0	0.5	0	1.5	1.0	1.5	2.0	3.5	3.0	3.5	3.0
$\hat{x}[k]$	0.5	0	1.5	1.0	1.5	2.0	3.5	3.0	3.5	3.0	0.5
$\hat{x}[k] - x[k]$	0.5	–0.3	0	0.3	0.5	–0.3	–0.2	0.2	0.1	0.2	0.5
Err. Direction Up/Down	U	D	–	U	U	D	D	U	–	U	U

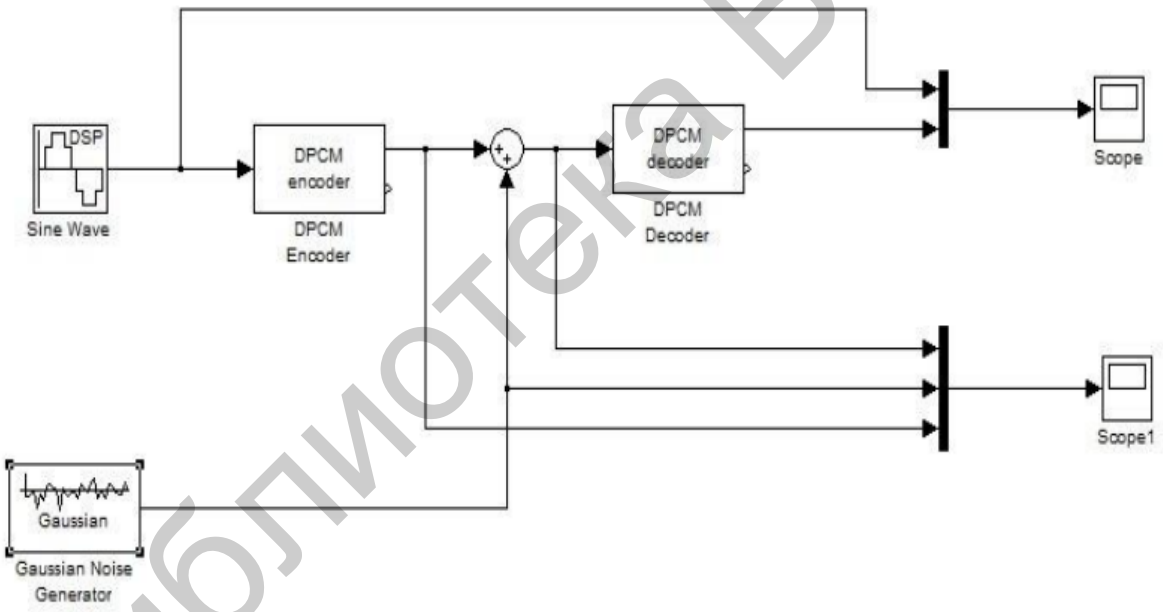


Рис. 32. Схема моделирования DPCM

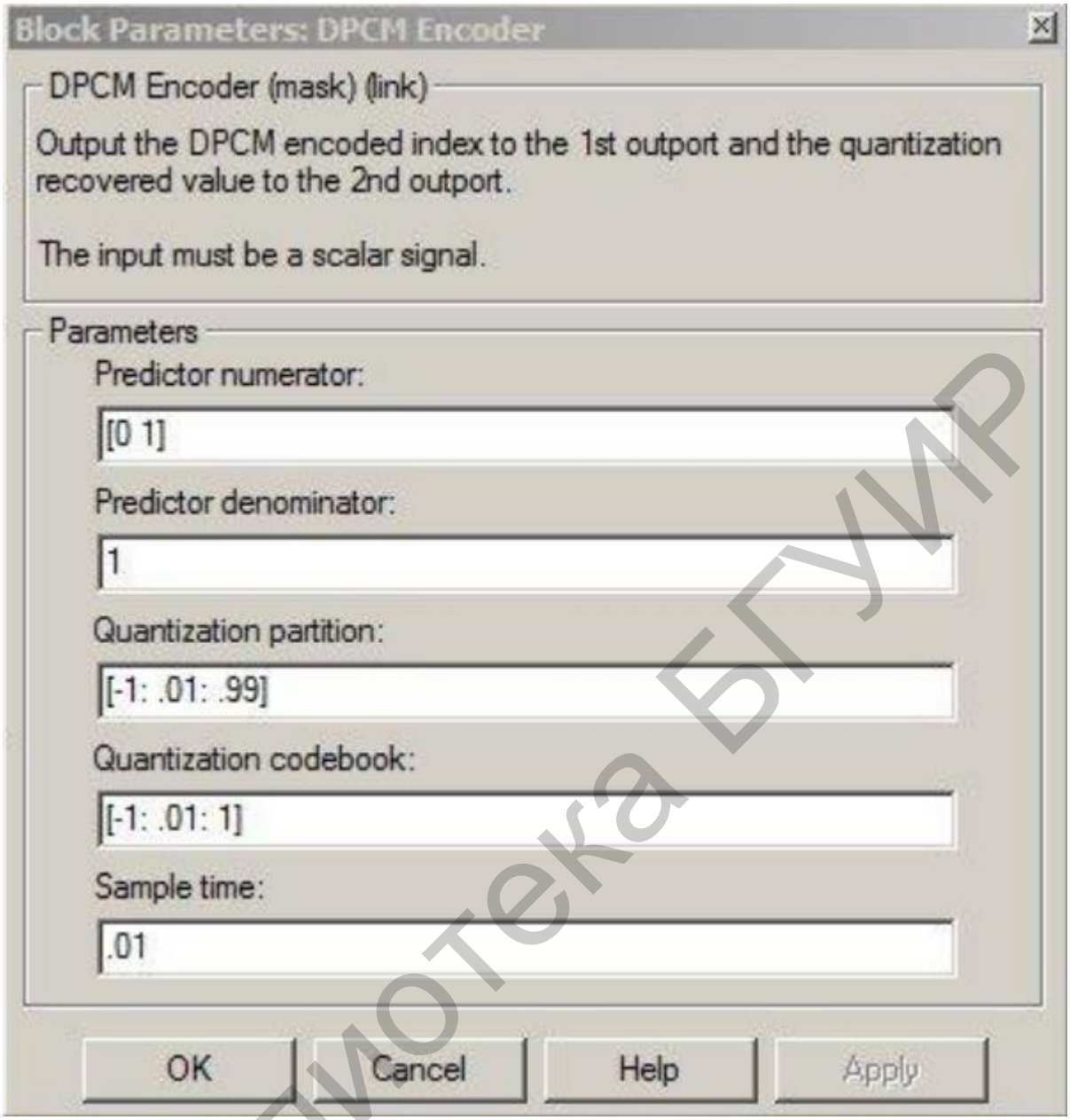


Рис. 33. Выбор параметров блока Encoder

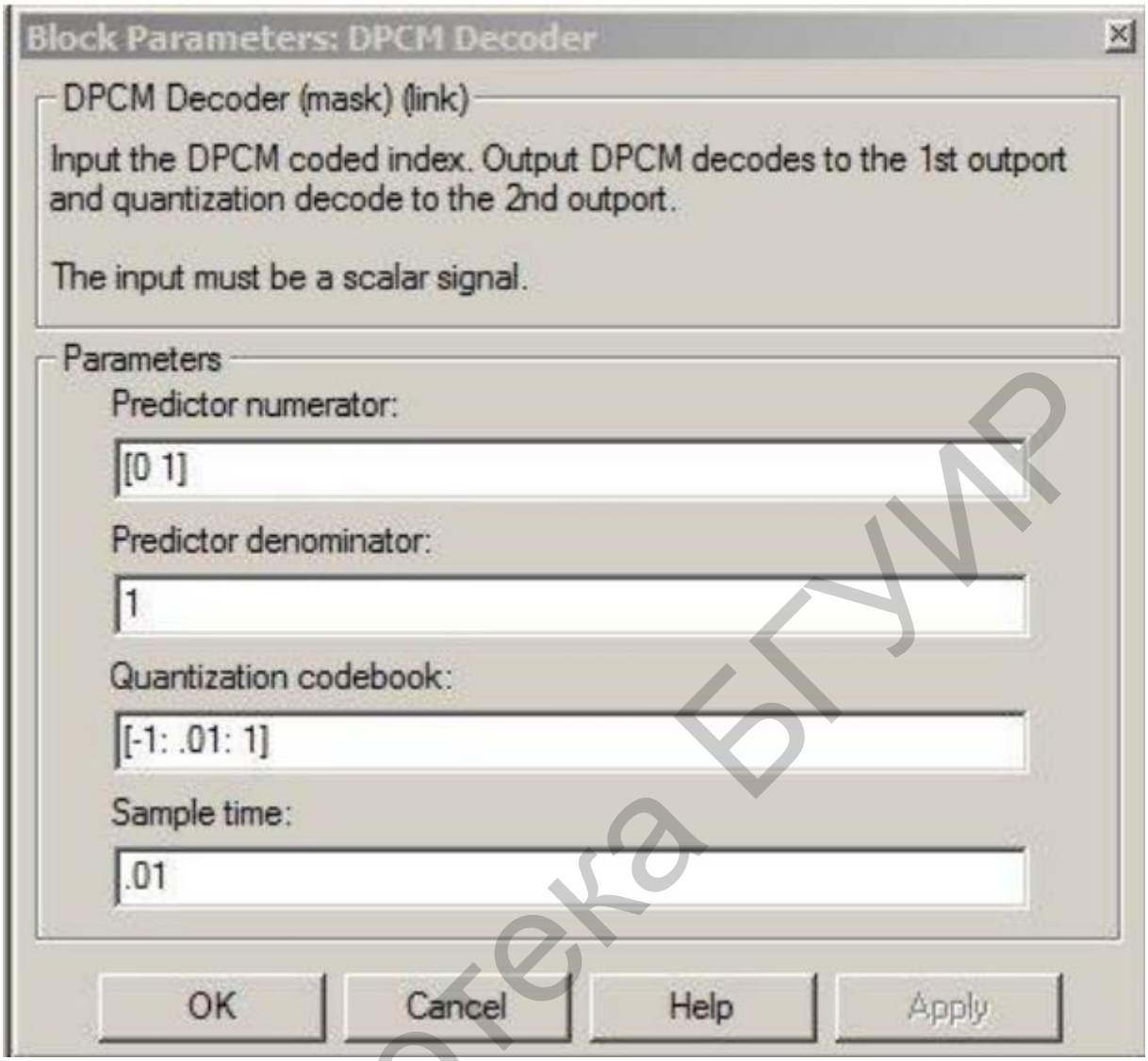


Рис. 34. Выбор параметров блока Decoder

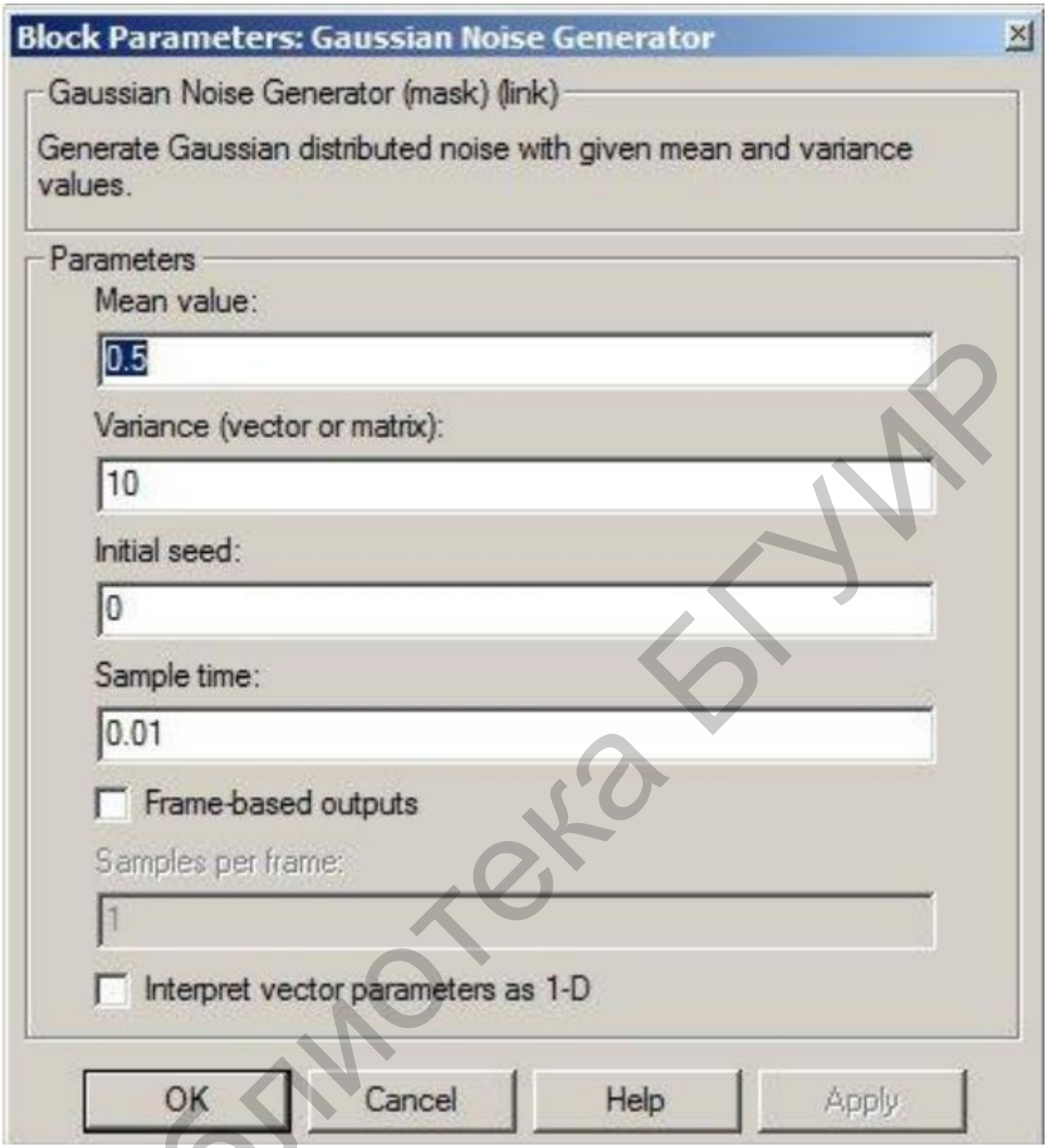


Рис. 35. Выбор параметров блока Gaussian Noise Generator

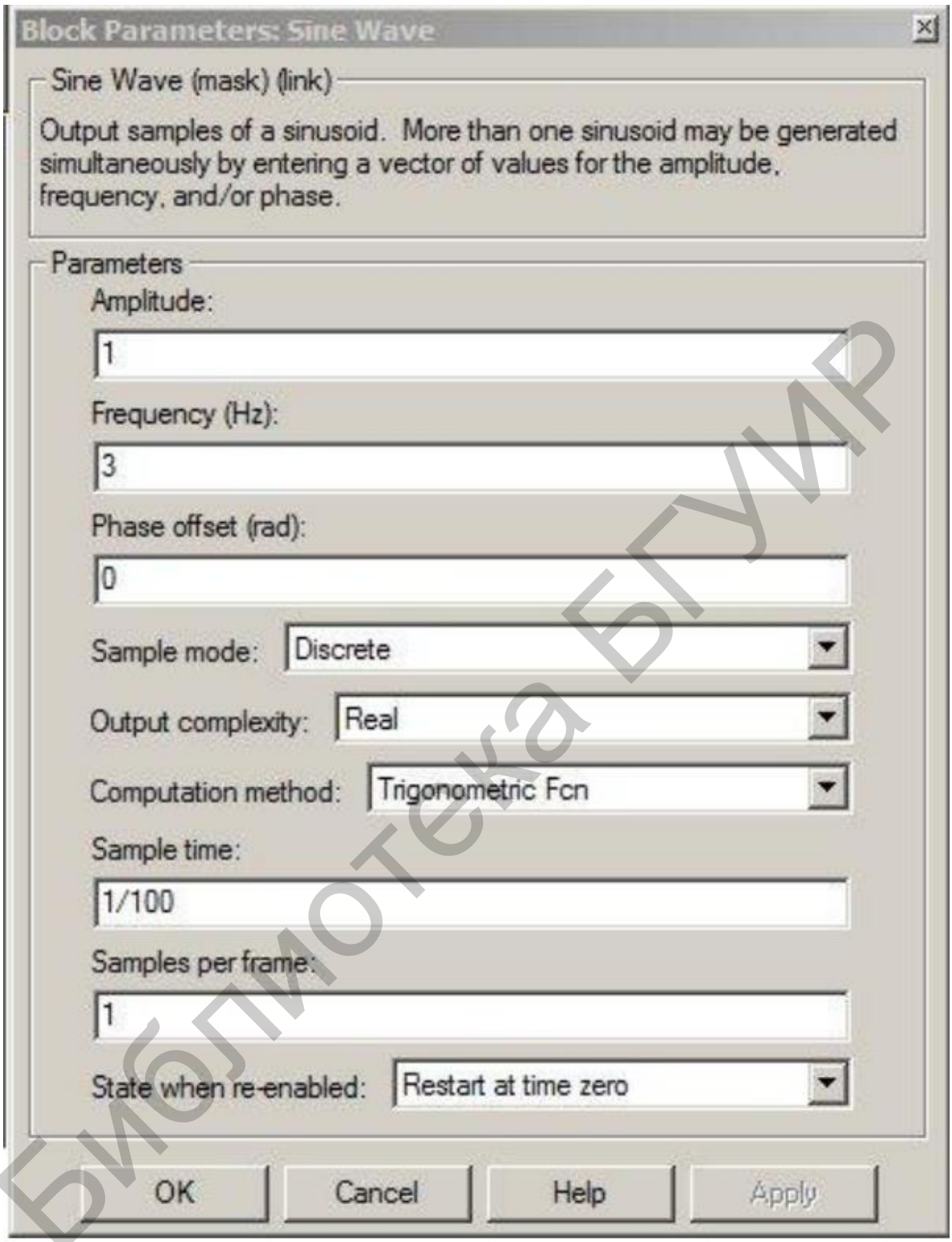


Рис. 36. Выбор параметров блока Sine Wave

Для блока Gaussian Noise Generator рекомендуемые значения показаны ниже:

- Menu value: mean value of noise, range from -1 to 1;
- Variance: random noise variance, range from 1 to 10;
- Initial seed^ initial state, leave 0;
- - Sample time^ sampling period, leave 0.01.

Можно использовать и другой генератор.

3. Контрольные вопросы

1. В чем состоит принцип получения импульсной амплитудной модуляции (Pulse Amplitude Modulation, PAM)?

2. Что представляет собой модуляция положением импульса (Pulse Position Modulation, PPM)?

3. Что представляет собой импульсная кодовая модуляция (Pulse Code Modulation, PCM)?

4. Что представляет собой избыточная информация в PCM?

5. Что такое дифференциальная импульсно-кодовая модуляция (Differential Pulse Code Modulation, DPCM)?

6. Что такое дифференциальная логарифмическая импульсно-кодовая модуляция?

7. В чем состоят причины обращения к логарифмической импульсно-кодовой модуляции?

8. Приведите эквивалентную форму неравномерного квантователя.

9. Что представляет собой дельта-модуляция?

10. В чем состоят преимущества дельта-модуляции по сравнению с PCM и DPCM?

11. Что представляет собой гранулированный шум?

12. Что представляет собой шум перегрузки по крутизне?

13. Какие методы адаптивной дельта-модуляции вам известны?

14. Назовите характерные особенности адаптивной дельта-модуляции.

15. Приведите структурные схемы кодера и декодера адаптивной дельта-модуляции.

16. Что представляет собой сигма-дельта модуляция?

17. Что представляет собой эффективное разрешение, описываемое эффективной разрядностью (Effective Number Of Bits, ENOB)?

18. Что понимается под разрешением АЦП?

19. В чем состоит линейная модель сигма-дельта модулятора?

20. В чем состоят смыслы функций передачи сигнала и шума в линейной модели сигма-дельта модулятора?

21. Что такое предельные циклы в сигма-дельта модуляторе?

Лабораторная работа №4
МОДЕЛИРОВАНИЕ СИГНАЛОВ С КВАДРАТУРНОЙ АМПЛИТУДНОЙ И ФАЗОВОЙ МАНИПУЛЯЦИЕЙ

1. Краткие теоретические сведения
1.1. Амплитудная манипуляция и фазовая манипуляция

Фундаментальная концепция цифровой радиосвязи – передавать информацию от передатчика к приемнику по аналоговым каналам связи с использованием несущих частот.

Говоря более специфическим языком, полоса передаваемых при этом частот включает процесс модуляции амплитуды, фазы или частоты аналоговой несущей частоты сигнала, благодаря которой сохраняется группа частот, занимаемая информационными сигналами перед тем, как они связываются с несущей частотой в процессе модуляции.

По определению, частота – это производная фазы по времени, поэтому можно расширить рассмотрение фазовой модуляции, включив в него и частотную модуляцию. Обычно несущая частота много выше символьной скорости модуляции, хотя это не всегда так. Во многих цифровых системах связи аналоговая несущая находится в области радиочастот, составляющих сотни и тысячи мегагерц, и переносит информацию со скоростью нескольких мегабод. В других системах несущая может находиться в области аудиочастот, символьная скорость которых составляет величину от нескольких сотен бод до нескольких тысяч бод.

Косинусоидальную несущую частоту f_c можно рассматривать как модулированную цифровым сигналом и потому представляющую сигнал

$$S(t) = A(t)\cos(2\pi f_c t + \theta(t)), \qquad (1)$$

где $A(t)$ – изменяющаяся во времени амплитуда сигнала; $\theta(t)$ – изменяющаяся во времени фаза сигнала. При цифровой фазовой манипуляции модулируют только фазу несущей, оставляя амплитуду неизменной.

Начнем обсуждение цифровой фазовой манипуляции с рассмотрения бинарной фазовой манипуляции (BPSK, Binary Phase Shift Keying), поскольку эта форма является простейшей формой цифровой фазовой манипуляции. Для BPSK каждый символ соответствует одному биту передаваемой информации. В соответствии с такой постановкой вопроса нужно выбрать два различных значения $\theta(t)$, одно из которых будет соответствовать логическому нулю, а другое – представлять логическую единицу. Так как цикл несущей составляет 2π радиан и символы могут иметь только два различных значения фазы несущей, то можно получить для передачи логических нуля и единицы выражения

$$S_0(t) = \sqrt{E_s}\cos(2\pi f_c t + \pi), \quad S_1(t) = \sqrt{E_s}\cos(2\pi f_c t + 0), \qquad (2)$$

где $\sqrt{E_s}$ – амплитуда модулированной косинусоидальной несущей.

Выражения, записанные для сигнала с фазовой манипуляцией $s(t)$, представленные соотношениями (2), описывают BPSK. Но если учесть, что $\cos(\theta+\pi) = -\cos\theta$, то соотношения (2) можно переписать как

$$S_0(t) = -\sqrt{E_s}\cos(2\pi f_c t), \qquad S_1(t) = +\sqrt{E_s}\cos(2\pi f_c t). \tag{3}$$

Эти выражения для BPSK показывают, что BPSK сигнал можно рассматривать как форму амплитудной модуляции с $A_0(t) = -1$ и $A_1(t) = 1$. Напрашивается вопрос: BPSK – это фазовая или амплитудная модуляция? Обе возможности рассмотрения этого процесса корректны, так как получены в результате выполнения корректных тригонометрических тождеств. Суть процесса модуляции легче понять, когда его рассматривать с точки зрения амплитудной модуляции.

Для рассматриваемого примера несущий сигнал – это $\sqrt{E_s}\cos(2\pi f_c t)$ и амплитуда модулирующего сигнала – импульс прямоугольной формы, имеющий амплитуду ± 1 и период T, длительность которого равна длительности одного символа.

1.2. Альтернативный выбор $\theta(t)$

Итак, было выбрано $\theta_0(t) = \pi$ и $\theta_1(t) = 0$. Но можно было выбрать и $\theta_0(t) = \pi/2$ и $\theta_1(t) = -\pi/2$. Тогда вместо (3) получилось бы

$$S_0(t) = -\sqrt{E_s}\cos(2\pi f_c t + \pi/2), \qquad S_1(t) = +\sqrt{E_s}\cos(2\pi f_c t - \pi/2) \tag{4}$$

Используя тождества $\sin\theta = \cos(\pi/2 - \theta)$, $\cos(-\theta) = \cos\theta$ и $\sin(-\theta) = -\sin\theta$, соотношения (4) можно переписать иначе:

$$S_0(t) = -\sqrt{E_s}\sin(2\pi f_c t), \qquad S_1(t) = +\sqrt{E_s}\sin(2\pi f_c t).$$

Результат снова представляет синусоидальную несущую, перемноженную с прямоугольным импульсом, соответствующим одному биту информации, хотя несущая теперь стала синусоидальной вместо косинусоидальной.

В этих примерах предполагалось, что длительность T каждого символа равнялась точно одному циклу несущей частоты или, иначе говоря, имелось две половины цикла несущей, приходящихся на длительность символа. Кроме того, передача символа имела место, когда фаза немодулированного сигнала равнялась нулю. Ни одна из этих особенностей не является необходимой. Например, можно выбрать символьную скорость, которая несоизмерима с несущей частотой. В этом случае передача символа будет иметь место при многих различных фазах несущей и длительность символа может быть иррациональной по отношению к длительности цикла несущей.

1.3. Формирование и демодуляция сигналов с квадратурной амплитудной модуляцией

Несмотря на многообразие многопозиционных сигналов, их формирование в передатчиках и демодуляция в приемниках проводится с использованием

общего технического решения, основанного на раздельном формировании двух независимых квадратурных составляющих модулирующего сигнала $I(t)$ и $Q(t)$ и их последующей передачи на одной несущей методом квадратурной амплитудной модуляции. Структурная схема модулятора сигнала с квадратурной амплитудной модуляцией приведена на рис. 1.

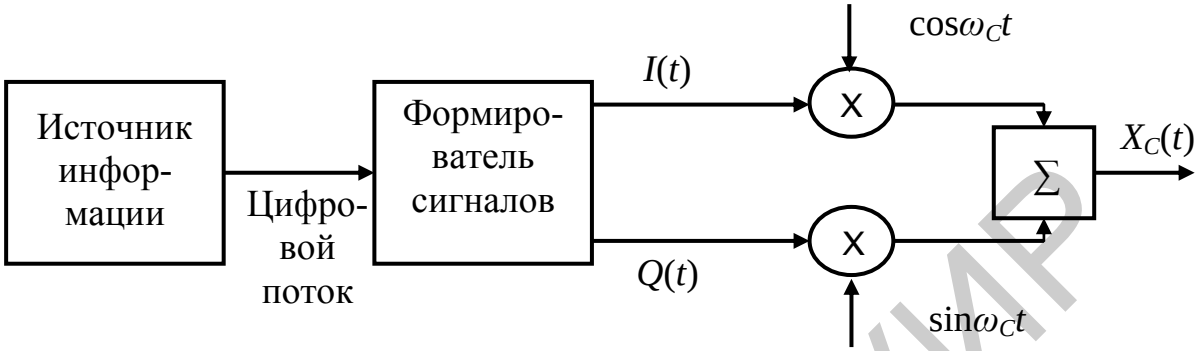


Рис. 1. Структурная схема квадратурного амплитудного модулятора

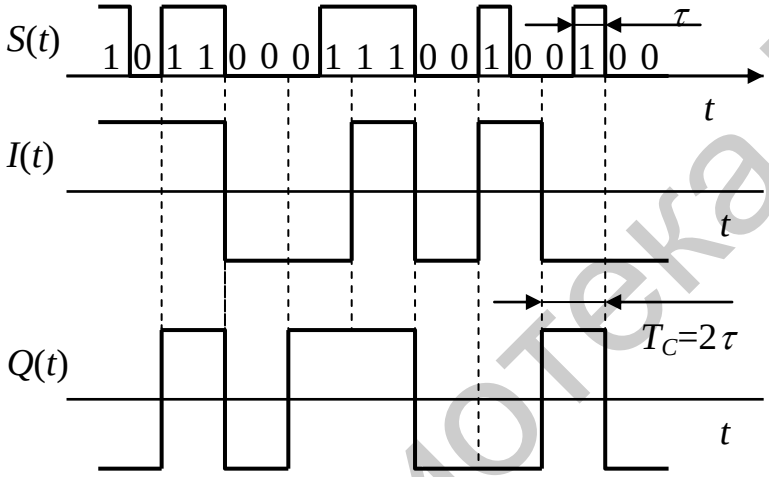


Рис. 2. Диаграммы, поясняющие процесс формирования сигнала ФМ-4

Формирователь сигналов с АИМ вырабатывает многоуровневые сигналы $I(t)$ и $Q(t)$, значения уровней которых равны проекциям векторов сигналов на вещественную и мнимую оси. Число уровней сигналов $I(t)$ и $Q(t)$ равны двум при ФМ-4, четырем – при АФМ-16, шести – при АФМ-64. Длительность символов равна длительности двух битов при ФМ-4, четырех битов – при АФМ-16, шести битов – при АФМ-64. Диаграммы, поясняющие процесс формирования сигнала ФМ-4, приведены на рис. 2, а сигнала АФМ-16 – на рис. 3.

На рис. 4 показана схема демодулятора многопозиционного сигнала. Демодулятор содержит два *синхронных детектора*, которые выделяют многоуровневые сигналы $I`(t)$ и $Q`(t)$. Из-за наличия в канале шумов они отличаются от переданных $I(t)$ и $Q(t)$. Задача решающего устройства – восстановить переданную цифровую последовательность с минимальным числом ошибок. Для нормальной работы синхронных детекторов частота несущей f_C должна быть точно равна частоте несущей передатчика. Это достигается с помощью специальных методов восстановления несущей.

Прием широкополосных ФМн-сигналов осуществляется оптимальным приемником, который для сигнала с полностью известными параметрами вычисляет корреляционный интеграл

$$Z = \int_0^T x(t)u(t)dt,$$

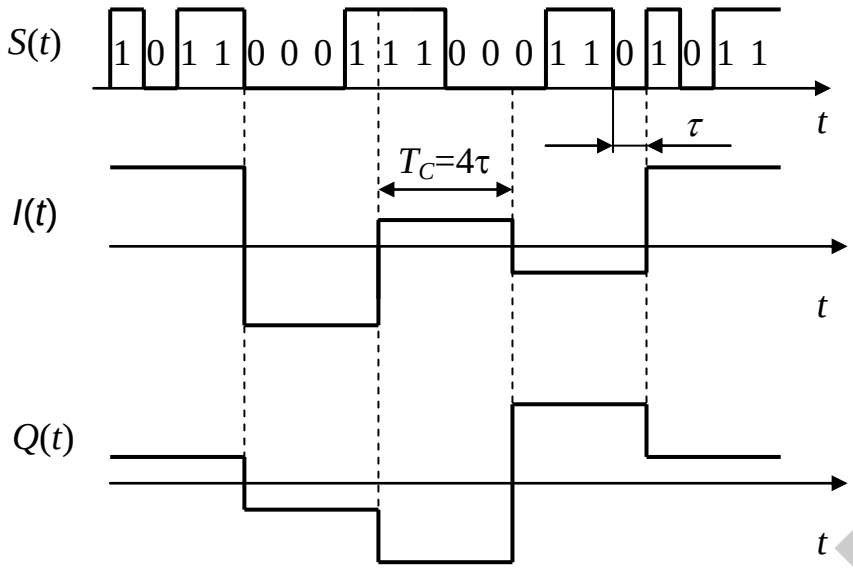


Рис. 3. Диаграммы, поясняющие процесс формирования сигнала АФМ-16

где $x(t)$ – входной сигнал, представляющий собой сумму полезного сигнала $u(t)$ и помехи, чаще всего являющейся аддитивным гауссовским белым шумом.

Затем величина Z сравнивается с порогом Z_0. Значение корреляционного интеграла находится с помощью согласованного фильтра или корреляторa.

Форма спектров всех упомянутых сигналов совпадает с формой спектра для сигнала ФМ-2, а ширина главного лепестка определяется тем же соотношением $\Delta f = 2/T_C$. Однако с учетом того, что длительность символа в случае многопозиционных сигналов превышает длительность бита, значения ширины полосы будут меньше: для сигналов ФМ-8 и АФМ-8 она составляет $2/3T_C$, для сигналов ФМ-16 и АФМ-16 – $1/2T_C$, для сигнала АФМ-64 – $1/3T_C$.

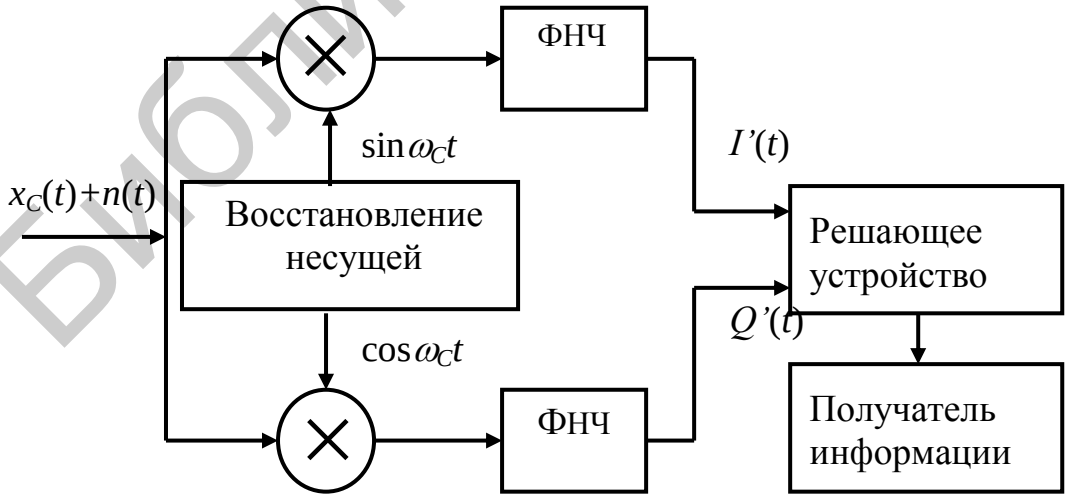


Рис. 4. Структурная схема демодулятора многопозиционного сигнала

1.4. Диаграммы состояний многопозиционных сигналов

При модуляции цифровым *биполярным NRZ-сигналом* на выходе *балансного модулятора* получается сигнал *двухпозиционной фазовой манипуляции* (*ФМ*-2). Спектр сигнала ФМ-2 образуется переносом спектра модулирующего NRZ-сигнала на высокочастотную *несущую* f_0. Ширина главного лепестка спектра при этом составляет $\Delta f = 2/T_C$, где T_C – длительность одного символа. С учетом того, что в случае *NRZ* модулирующего сигнала длительность символа совпадает с длительностью одного бита $T_Б$, можно записать

$$\Delta f = 2/T_Б.$$

Таким образом, обладая высокой помехоустойчивостью, ФМ-2 характеризуется низкой спектральной эффективностью.

Большей спектральной эффективностью обладают многопозиционные сигналы, из которых наиболее часто используют:

- четырехпозиционную фазовую манипуляцию (ФМ-4),
- шестнадцатипозиционную амплитудно-фазовую манипуляцию (АМФ-16).

Используют также сигналы ФМ-8, АФМ-8, АФМ-64 и другие. В многопозиционных сигналах каждый импульс несет информацию о нескольких битах: о двух битах в случае ФМ-4, о трех – при ФМ-8 и АФМ-8, о 6 – при АФМ-64. Сигнальные созвездия для сигналов ФМ-4 и АФМ-16 приведены на рис. 5, а и рис.7.

Каждое из четырех возможных состояний сигнала ФМ-4 отличается фазовым сдвигом. Амплитуда же сигнала постоянна.

Состояния же сигнала АФМ-16 различаются как фазовыми сдвигами, так и амплитудами.

Постоянство огибающей при ФМ-2, ФМ-4, ФМ-8 является важным преимуществом этих сигналов, так как высокочастотные тракты передатчика и приемника при этом могут работать в нелинейном режиме, что позволяет получить максимальную выходную мощность при высоком КПД передатчика.

При использовании АФМ-8, АФМ-16 и АФМ-64 высокочастотные тракты передатчика и приемника должны быть высоколинейными. Энергетические показатели передатчика при этом существенно хуже, чем при использовании сигналов с постоянной огибающей.

1.5. Помехоустойчивость многопозиционных сигналов

Помехоустойчивость различных видов модуляции удобно связать с минимальным расстоянием между возможными состояниями сигнала d_{min}, выраженную через среднюю энергию, приходящуюся на один бит E_b.

В случае ФМ-2 $d_{min} = 2A = 2\sqrt{E_b}$, где A – амплитуда сигнала.

При ФМ-4 $d_{min} = \sqrt{2}A = \sqrt{2}\sqrt{2}E_b = 2\sqrt{E_b}$.

Таким образом, при ФМ-4 минимальное расстояние между сигналами такое же, как и при ФМ-2. Следовательно, ФМ-4 обладает такой же помехоустой-

чивостью, что и ФМ-2. Занимаемая же полоса частот при ФМ-4 в два раза уже. Именно поэтому в действующих системах связи предпочтение отдают ФМ-4.

Остальные виды модуляции имеют меньшую помехоустойчивость, чем ФМ-2 и ФМ-4. Значения d_{min} для различных видов модуляции приведены ниже:

ФМ-8 $1{,}325\sqrt{E_{b\,cp}}$, АФМ-8 $1{,}549\sqrt{E_{b\,cp}}$,

ФМ-16 $0{,}78\sqrt{E_{b\,cp}}$, АФМ-16 $1{,}265\sqrt{E_{b\,cp}}$.

Из сравнения ФМ-8 и АФМ-8 следует, что при одинаковой полосе АФМ обладает большей помехоустойчивостью. Еще в большей степени это относится к ФМ-16 и АФМ-16. Именно поэтому ФМ-16 практически не используют, а АФМ-16 распространена очень широко.

1.6. Модуляция QPSK и π/4-DQPSK

Модуляция QPSK (Quadrature или Quaternary Phase Shift Keying)– квадратурная фазовая модуляция является решением компромисса между скоростью передачи и помехоустойчивостью и применяется как самостоятельно, так и в комбинациях с другими методами. Диаграмма состояний модуляции QPSK и дифференциальной QPSK со смещением на π/4 (π/4-DPSK) показана на рис. 5. При реализации дифференциального кодирования в сочетании со сдвигом несущей на $\pi/4$ сигнальное созвездие формируется двумя четырехточечными созвездиями QPSK, наложенными со сдвигом 45^0. В результате в сигнале присутствуют восемь фазовых сдвигов, причем фазы символов выбираются поочередно, то из одного созвездия, то из другого. Последовательные символы имеют относительные фазовые сдвиги, соответствующие одному из четырех углов: $\pm\pi/4$ и $\pm 3\pi/4$.

В технологии π/4-DQPSK все импульсы информационной последовательности модулятора разбиваются на пары, представляющие собой *2*-битовые символы, и при переходе от одной пары к другой начальная фаза несущей частоты сигнала изменяется на величину $\Delta\varphi$, которая определяется битами символа в соответствии с алгоритмом, приведенным выше.

Кружочками обозначены дискретные значения, которые может принимать фаза несущей, отсчитываемая от некоторого начального значения. Стрелками указаны возможные переходы между разрешенными значениями фазы. Оси координат соответствуют синфазной (Imphase – I) и квадратурной (Quadrature – Q) составляющим сигнала. Мы видим, что фазовая диаграмма сигнала π/4DQPSK состоит фактически из двух диаграмм квадратурной фазовой манипуляции (QPSK): фазовые состояния одной из них помечены значком ⊕, а другой – значком ⊗, и диаграммы сдвинуты одна относительно другой на угол π/4.

При переходе от одного символа к другому происходит изменение фазы от одного состояния первой диаграммы к одному из возможных состояний второй, а при переходе к следующему символу – возврат к предыдущей диаграмме, хотя скорее всего не к прежнему фазовому состоянию.

Результирующий выходной сигнал модулятора можно представить в виде $s(t)=\cos(\omega_0 t+\varphi_k)$, где ω_0 – несущая частота, $\varphi_k = \varphi_{k-1} + \Delta\varphi$ – начальная фаза на интервале k-го импульса.

Структурная схема модулятора показана на рис. 6. Входной поток данных D разделяется на два параллельных потока A и B, которые затем в преобразователе кода перекодируются в относительный код двух каналов (компонентов) I' и Q'.

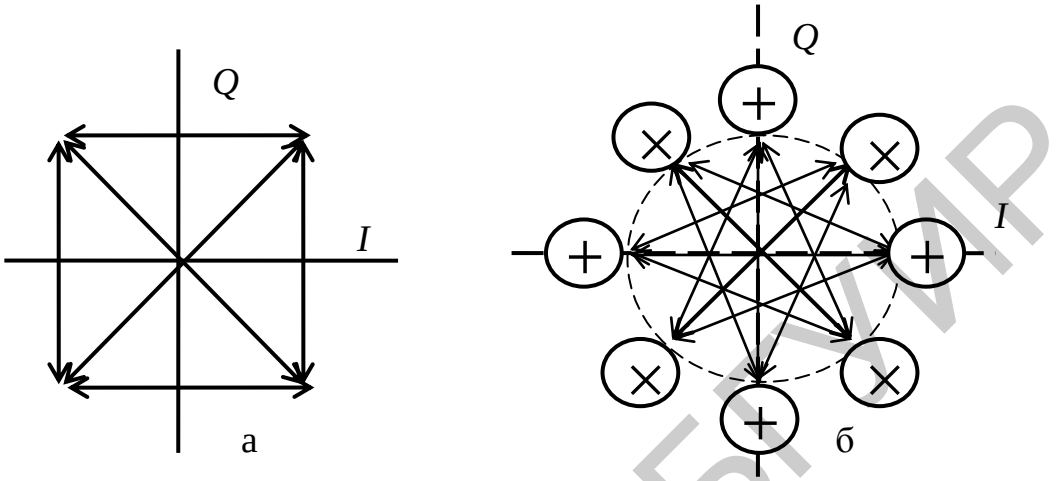


Рис. 5. Диаграммы состояний сигналов QPSK (а) и π/4DQPSK (б)

Правило кодирования Таблица 1

Биты входной последовательности модулятора		Изменение фазы $\Delta\varphi$
1	1	-3π/4
0	1	3π/4
1	0	-π/4
0	0	π/4

Правило кодирования изменения фазы Δφ битами входной последовательности представлено в табл. 1

Цифровые потоки I' и Q' подвергаются сглаживанию в формирующих фильтрах (ФФ), выходные сигналы которых I и Q непосредственно управляют работой четырехфазового модулятора, состоящего из двух балансных модуляторов и сумматора. Фазовый сдвиг несущих в каналах I и Q равен 90^0. Правило кодирования фазовых сдвигов соответствует выполнению условий (0, 0) – 45^0, (0,1) – 135^0, (1,0) – 315^0, (1,1) – 225^0.

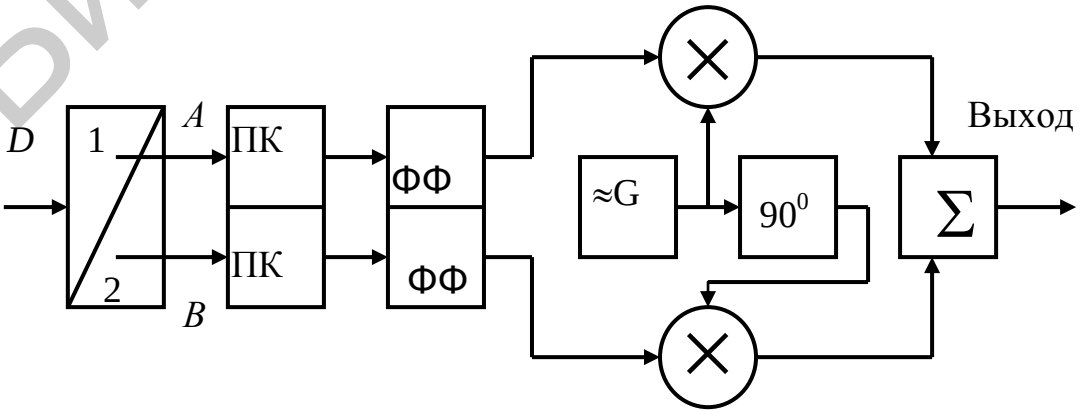


Рис. 6. Структурная схема модулятора QPSK

1.7. Глазковая диаграмма

Глазковая диаграмма – это суммарный вид всех битовых периодов измеряемого сигнала, наложенных друг на друга. Другими словами, изображение сигнала от начала периода 2 до начала периода 3 накладывается на изображение сигнала от начала периода 1 до начала периода 2, и так далее для всех битовых периодов. Типичная глазковая диаграмма характеризуется достаточно ровными и симметричными и плавными переходами (правая и левая точки пересечения), большим широко открытым «глазом», предоставляющим место для точной идентификации бита. Если пробная точка расположена в центре глаза, где сигнал достигает своего максимума или минимума, то очень маловероятно возникновение битовой ошибки. Расстояние между левой и правой точками пересечения называют единичным интервалом (unit interval).

Глазковая диаграмма не просто предоставляет множество информации, она удобна простотой применения и тем, что может применяться для измерений в любой цепи с реальными данными. Для глазковой диаграммы не требуется наличие особого тестового сигнала, хотя при желании можно использовать измерительный сигнал импульсного генератора. Она может эффективно применяться при исследовании случайных и псевдослучайных данных и относится к диапазонным измерениям.

Следствием формирования импульсов является необходимость точного представления изменения во времени символа, поступающего на вход приемника. Импульсы с косинусоидальным сглаживанием трудно идентифицировать, так как они изменяются плавно и постепенно. Глазковые диаграммы обеспечивают простой способ наблюдения прохождения между символами и контролирование изменения символа во времени. Глазковая диаграмма – это просто повторение частот сигнала в виде группы частот, занимаемых им перед тем, как он был связан с несущей в процессе модуляции на интервале одного символа. Максимально открытый глаз указывает на центр символа, который также представляет оптимальное время для приемника, чтобы принять отсчет. Переход между символами заставляет глаз закрываться.

Рис. 14 показывает глазковую диаграмму для сигнала с квадратурной амплитудной манипуляцией. В центре символа глаз сходится в двух точках: ± 1, в то время как на краях символа глаз принимает много различных значений. Точная форма глаза определяется числом отсчетов, приходящихся на символ.

2. Подготовка к выполнению лабораторной работы

1. По документации к пакету MATLAB ознакомьтесь с функциями, с помощью которых выполняется моделирование сигналов с квадратурной модуляцией и квадратурной манипуляцией.

2. Постройте спектрограммы спектров сигналов с квадратурной амплитудной модуляцией и квадратурной амплитудной манипуляцией, когда в каче-

стве модулирующего колебания используются монохроматические сигналы различных или одинаковых частот.

3. Постройте созвездия сигналов с квадратурной модуляцией.

3. Порядок выполнения лабораторной работы

3.1. Формирование сигнала с квадратурной амплитудной модуляцией

3.1.1. Амплитудная манипуляция

Амплитудная манипуляция (Amplitude shift keying, ASK) представляет собой частный случай амплитудной модуляции, когда амплитуда несущего колебания скачкообразно изменяется. Поэтому сначала рассмотрим построение временной диаграммы (осциллограммы) амплитудно-манипулированного сигнала.

Пример 1. Построить временную диаграмму амплитудно-манипулированного сигнала, несущая частота *Fc* модулируется низкочастотным сигналом, имеющим четыре уровня: 1, 2, 3, 4.

Решение: Составим последовательность кодов системы MATLAB.

```
sy = [1 3 2 4 1 3 2 4 2]; %передаваемые (модулирующие) символы
Fd = 1; % примем единичную символьную скорость передачи данных
Fc = 4; % несущая частота
FsFd = 40; % отношение несущей частоты к частоте дискретизации
Fs = Fd*FsFd; % частота дискретизации
t = (0:length(sy)*FsFd-1)/Fs; % установим дискретное время
% переходим к формировнию сигнала с амплитудной манипуляцией
s_ASK = sy(floor(Fd*t)+1).*cos(2*pi*Fc*t);
plot(t,s_ASK)
xlabel('Время');
ylabel('Амплитуда ASK-сигнала');
ylim([-4.1 4.1]);
title('Амплитудно-манипулированый сигнал');
```

Результат работы этой программы показан на рис. 7.

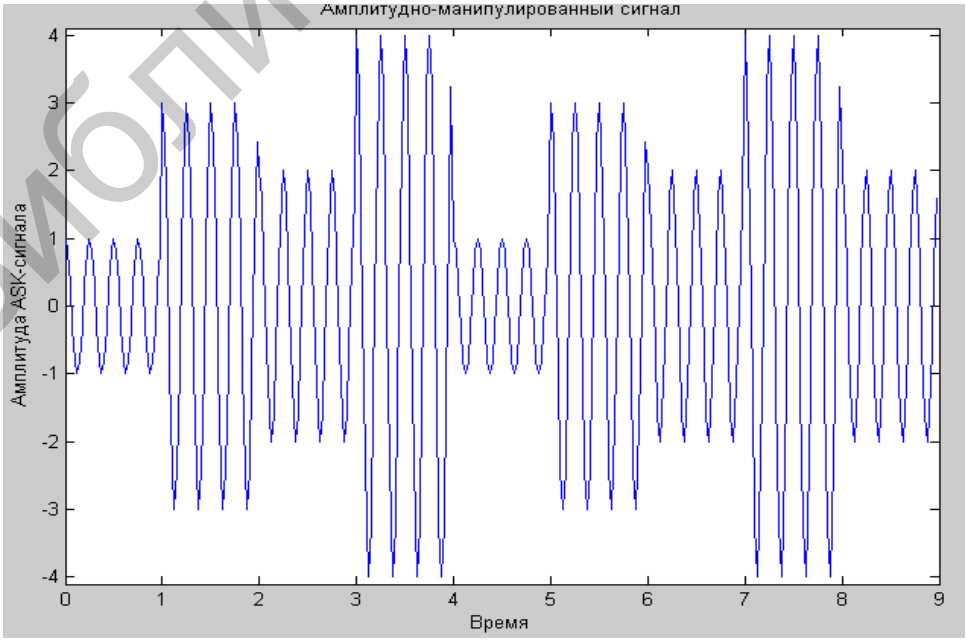


Рис. 7. Временная диаграмма амплитудно-манипулированного сигнала

3.1.2. Сигнал с квадратурной амплитудной манипуляцией

Перейдем теперь к построению сигнала с квадратурной аплитудной манипуляцией.

Пример 2. Построить начальный участок временной диаграммы сигнала с аплитудной манипуляцией, соответстующий первым 100 модулирующим символам, выбранным случайным образом из множества возможных значений {–3, –1, 1, 3}.

Решение:

Для построения такого графика временной диаграммы амплитудно-манипулированного сигнала выберем число символов, равное N = 1000, и сформируем значения символов с помощью функции randint с ограниченным верхним пределом, чтобы получился удобный для интерпретации вид сигнала с квадратурной амплитудной манипуляцией. Поэтому создадим векторы амплитуд синфазной и квадратурной составляющих, значения которых случайно выбраны из набора {–3, –1, 1, 3}.

```
N=1000;
aa=randint(1,N,4);
bb=randint(1,N,4);
a1=2*aa-3;
% преобразуем значения вектора aa к требуемому набору
b1=2*bb-3;
% преобразуем значения вектора bb к требуемому набору
Fd=2400;
% установим символьную скорость, равную 2400 символ/с
Fc=1800;
% установим несущую, равную 1800 Гц
FsFd=4;
% положим число отсчетов на формирование одного символа, равное 4
Fs=Fd*FsFd;
% установим частоту дискретизации, равную 9600 Гц
a1=repmat(a1,FsFd,1);
% продублируем каждый отсчет 4 раза
a1=a1(:);
b1=repmat(b1,FsFd,1);
b1=b1(:);
t=(0:N*FsFd-1)/Fs;
% сформируем аналоговый сигнал, чтобы продемонстрировать
% форму сигнала с квадратурной амплитудной манипуляцией
t=t';
s_qask16=a1.*cos(2*pi*Fc*t)+b1.*sin(2*pi*Fc*t);
figure(1)
figure(20)
figure(20)
plot(t(1:100),s_qask16(1:100))
xlabel('Time')
ylabel('16QAM Signal')
ylim([-5 5])
```

Полученная временная диаграмма приведена на рис. 8.

Задание 1. Для других параметров повторите выполнение данного раздела этой работы и отметьте характерные отличия, наблюдающиеся при этом.

Параметры сформированного сигнала, т. е. вид созвездия, значения символьной скорости и несущей частоты соответствуют модему, передающему данные со

скоростью (600 бит/с в соответствии с Рекомендацией ITU-T V.32. Прослушаем сигнал, используя для этого функцию soundsc, чтобы не заботиться о приведении сигнала к диапазону уровней {–1 … 1}. Для этого достаточно выполнить команду

```
soundsc(repmat(s_qask16, 10, 1),Fs)
```

Функция repmat здесь используется для повторения сформированного сигнала 10 раз, чтобы увеличить длительность звучания сигнала.

При квадратурной манипуляции могут меняться и амплитуда и фаза несущего колебания, поэтому амплитудная и фазовая манипуляции представляют собой частные случаи квадратурной модуляции и для получения квадратурной манипуляции используют созвездия амплитудной и фазовой манипуляции.

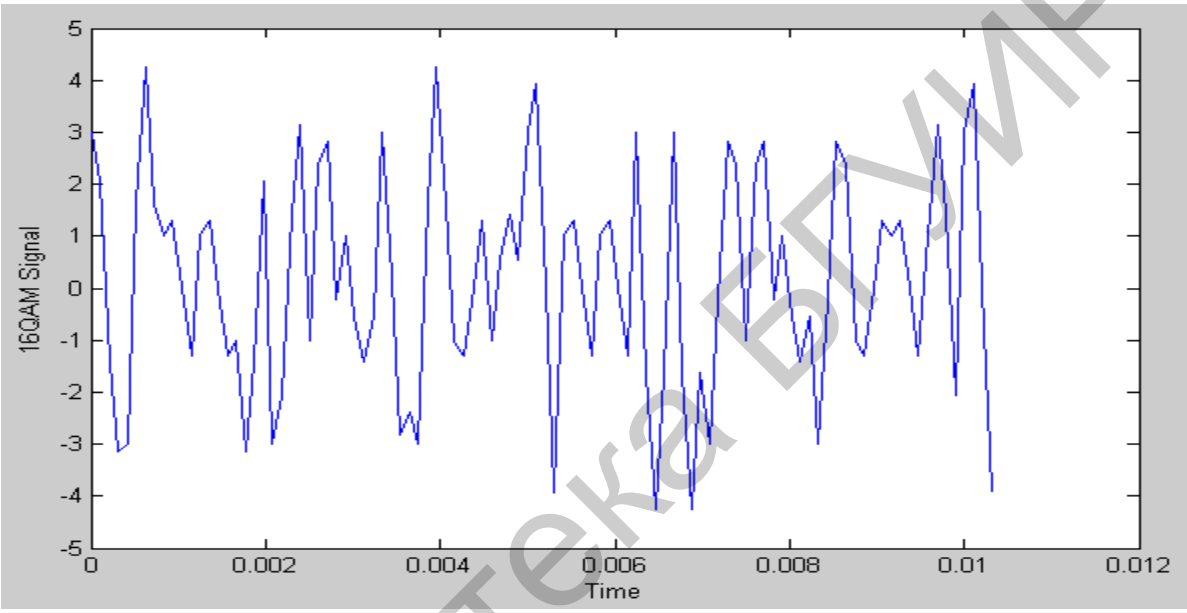


Рис. 8. Сигнал с 16-позиционной квадратурной манипуляцией

Пример 3. Построить графики созвездий, соответствующих 8-позиционной амплитудной манипуляции и 8-позиционной фазовой манипуляции.

Решение:

Покажем графики этих созвездий на рис. 9, которые можно получить с помощью следующего кода.

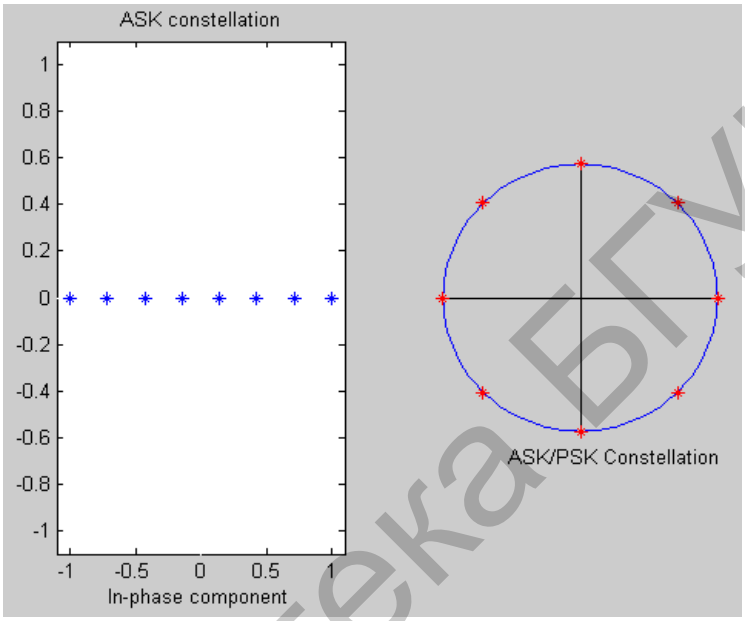
```
figure(22)
subplot(1,2,1)
modmap('ask',8)
subplot(1,2,2)
modmap('psk',8)
```

Задание 2. Задайте другие параметры, отличающиеся от приведенных в примере 3, и постройте новые формы созвездий, полученных при этом.

3.2. Демодуляция квадратурно-манипулированного сигнала

Рассмотрим процесс деманипуляции сигнала с квадратурной манипуляцией, полученный в п. 3.1 настоящей работы.

При выполнении деманипуляции квадратурной манипуляции демодулируемый сигнал умножается на два несущих колебания, сдвинутых по фазе друг относительно друга на 90^0, а результаты умножения пропускаются через соответствующие ФНЧ. На выходе этих фильтров получаются аналоговые сигналы синфазной и квадратурной составляющих. Далее эти сигналы дискретизируются с частотой, равной символьной скорости. Пары отсчетов синфазной и квадратурной составляющих образуют комплексное число, и ближайшая к этому числу точка используемого созвездия (а точнее – соответствующий этой точке информационный символ) выдается в качестве выходного результата.



а б

Рис. 9. Созвездия, соответствующие 8-позиционной амплитудной (а) и 8-позиционной фазовой манипуляции (б)

Реализуем описанный алгоритм деманипуляции для сформированного ранее сигнала *s_qask16*.

Пример 4. Приведенный ниже код реализует квадратурную деманипуляцию, дискретизацию полученного сигнала с символьной частотой, вывод графика расположения принятых точек на комплексной плоскости, называемый диаграммой рассеяния (scatter plot) и показанной на рис. 10, выбор ближайших координат точек из использованного созвездия и сравнение полученных синфазных *a2* и квадратурных *b2* амплитуд с исходными амплитудами *aa* и *bb*.

```
y=s_qask16.*exp(j*2*pi*Fc*t)*2;
[b,a]=butter(2,Fd*2/Fs);
y=filtfilt(b,a,y);
z=y(3:FsFd:end);
figure(25)
plot(z,'.')
axis square
a2=round((real(z)+3)/2);
a2(find(a2<0))=0;
a2(find(a2>3))=3;
b2=round((imag(z)+3)/2);
```

```
b2(find(b2<0))=0;
b2(find(b2>3))=3;
symerr(aa',a2)
symerr(bb',b2)
```

Функция round применяется здесь для поиска точки используемого созвездия, наиболее близкой к принятому отсчету сигнала. Такое решение возможно благодаря простой структуре использованного нами созвездия. В более общем случае для этого необходимо воспользоваться функцией demodmap.

Функция symerr подсчитывает число неправильно принятых символов, так как она возвращает число различающихся элементов двух векторов одинаковой длины. Как видим по результатам применения этой функции, сигнал принят без ошибок.

Задание 3. Запустите на выполнение эту программу в виде функции и проверьте ее работоспособность при других значениях вводимых данных. Результаты выполненных измерений занесите в отчет по данной работе.

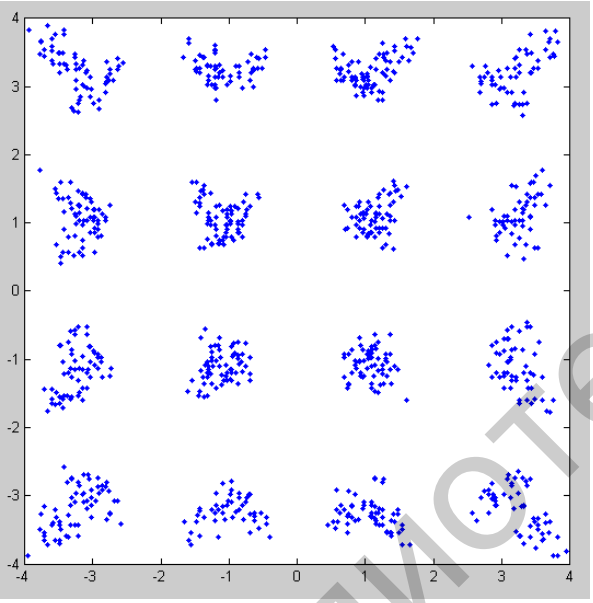


Рис. 10. Диаграмма рассеяния при приеме сигнала с квадратурной манипуляцией

Сформируем спектр полученного в п. 3.1 настоящей работы сигнала с квадратурной амплитудной манипуляцией.

Если параметры манипуляции аналогового сигнала поддерживаются постоянными в течение символьного такта и в начале нового такта изменяются скачкообразно, то это приводит к появлению скачков в сформированном сигнале. Чтобы сделать спектр более компактным, необходимо обеспечить гладкость изменения сигнала, а это в свою очередь означает, что модулирующая функция тоже должна быть гладкой. Следовательно, вместо скачкообразного изменения параметров модуляции необходимо выполнить интерполяцию между точками созвездия, соответствующими последовательным символам.

В соответствии с теоремой Котельникова можно соединить отсчеты, следующие с символьной скоростью `*Fd* плавной функцией, занимающей полосу частот от нуля до *Fd*/2. В этом случае квадратурно-манипулированный сигнал будет занимать полосу частот шириной *Fd*. Однако медленное затухание функ-

ций sinc, составляющих базис Котельникова, делает неудобной интерполяцию на их основе. Наибольшее распространение при интерполяции отсчетов для цифровой модуляции получил SQRT-вариант фильтра с косинусоидальным сглаживанием амплитудно-частотной характеристики (square root-cosine filter).

Фильтр, используемый для интерполяции, определяет форму спектра квадратурно-манипулированного сигнала. Поэтому его называют формирующим фильтром (shaping filter), а сам процесс интерполяции – формированием спектра (spectral shaping).

Пример 5. Выполнить скачкообразное изменение параметров модуляции, которое можно рассматривать как использование формирующего фильтра с прямоугольной импульсной характеристикой, длительность которой равна символьному интервалу.

Решение:

Сначала сформируем снова 16-позиционный квадратурно-манипулированный сигнал, используя на этот раз формирующий фильтр с косинусоидальным сглаживанием амплитудно-частотной характеристики.

```
a1s=2*aa-3;
b1s=2*bb-3;
a1s=rcosflt(a1s,Fd,Fs,'sqrt');
b1s=rcosflt(b1s,Fd,Fs,'sqrt');
t=(0:length(a1s)-1)/Fs;
t=t';
s_qask16s=a1s.*cos(2*pi*Fc*t)+b1s.*sin(2*pi*Fc*t);
figure(26)
plot(t(1:100),s_qask16s(1:100))
xlabel('Time');
ylabel('Signal 16QAM after Filtering')
ylim([-2.5 2.5])
```

Форма сформированного сигнала с квадратурной манипуляцией показана на рис. 11.

Используя функцию soundsc(repmat(s_qask16s,10,1),Fs), прослушаем этот сигнал:

soundsc(repmat(s_qask16s,10,1),Fs)

Звук стал больше похож на тот, что производится модемом, так как все дело именно в формировании спектра этого сигнала.

Теперь сравним спектры мощности сигналов s_qask16 и s_qask16s, чтобы наглядно убедиться во влиянии формирующего фильтра. Для этого выполним команды пакета MATLAB:

```
[P1, f] = pwelch(s_qask16, [], [], [], Fs);
P2 = pwelch(s_qask16s, [], [], [], Fs);
plot(f,[P1 P2]);
```

Задание 4. Оформите записанную в примере 5 программу в виде функции и проверьте ее выполнимость, задав сначала те же параметры, при которых эта программа была выполнена, а затем подставив в нее другие параметры.

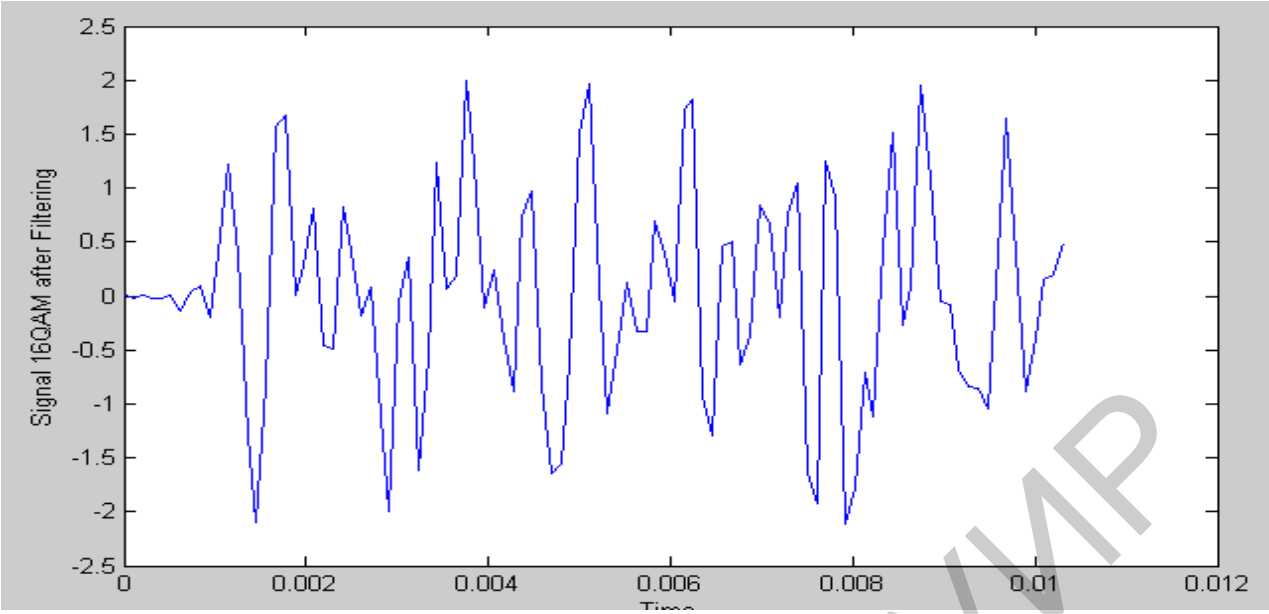


Рис. 11. Сигнал с 16-позиционной квадратурной манипуляцией при использовании формирования спектра

Из графиков, помещенных на рис. 12 и 13, видно, что при использовании формирующего фильтра спектр сигнала оказывается значительно компактнее. При приеме такого сигнала в качестве ФНЧ нужно использовать такой же фильтр, как для формирования спектра. Последовательное использование двух SQRT-фильтров с косинусоидальным сглаживанием дает результирующую импульсную характеристику, равную нулю в точках, сдвинутых на целое число символов относительно центрального пика. Это позволяет при правильном выборе моментов взятия отсчетов устранить помехи от соседних символов, так называемую межсимвольную интерференцию (intersymbol interference, ISI).

Для получения глазковой диаграммы и интерпретации результатов приема сигнала s_qask16s умножим последний на опорное комплексное колебание и подготовим данные для глазковой диаграммы.

```
y=s_qask16s.*exp(j*2*pi*Fc*t)*2;
b=rcosine(Fd,Fs,'sqrt');
y=filter(b,1,y);
h=eyediagram(y,8,10,0);
```

Глазковая диаграмма приведена на рис. 14. Построим диаграмму рассеяния, полученную при приеме сигнала с квадратурной манипуляцией.

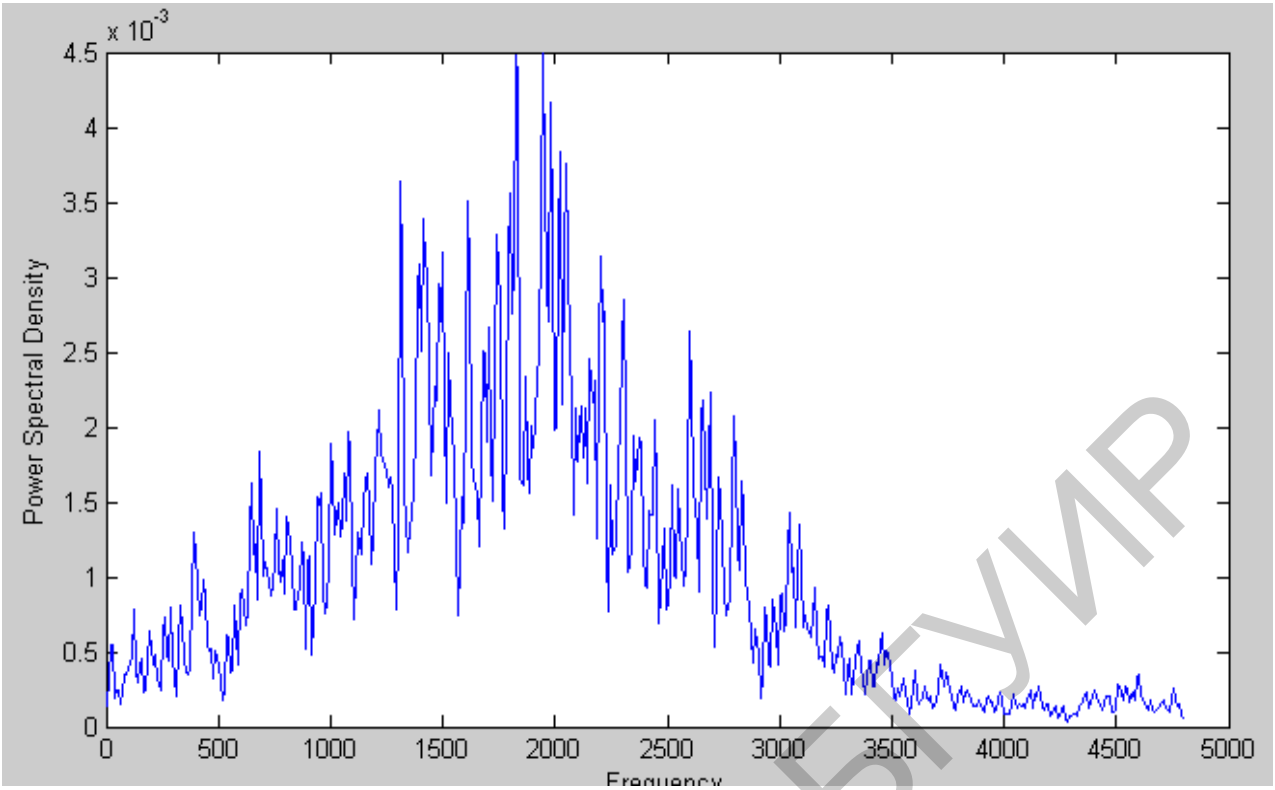


Рис. 12. Спектр плотности мощности сигнала
с квадратурной манипуляцией при отсутствии формирующего фильтра

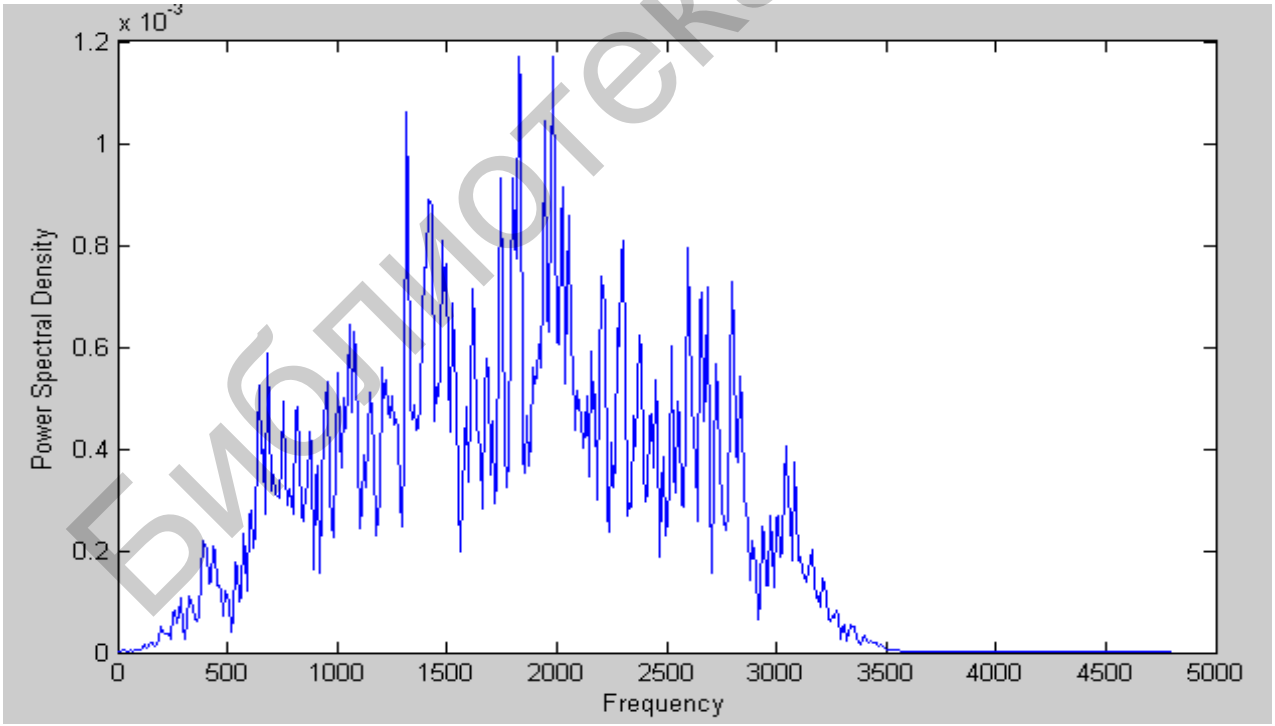


Рис. 13. Спектр плотности мощности сигнала
с квадратурной манипуляцией при наличии формирующего фильтра

Глазковая диаграмма – это своего рода «осциллограмма» сигнала, полученная при длительности «прямого хода развертки», равной одному символьному такту, и бесконечном «времени послесвечения экрана». В точках оптимальной дискретизации линии на такой диаграмме образуют узкие пучки, свободное пространство между которыми по форме напоминает раскрытый глаз. В данном случае видно, что выбирать элементы из вектора *y* нужно начиная с первого. Поскольку сигнал является комплексным, то приведены отдельные графики для его вещественной и мнимой частей.

Диаграмма рассеяния показана на рис. 15.

Выполним следующий код программы MATLAB:

```
figure(36)
z=y(1:FsFd:end);
z(1:6);% удаление «начального хвоста»
plot(z,'.')
```

Благодаря использованию согласованных друг с другом фильтров, на передающей и приемной сторонах разброс точек оказывается значительно меньше, чем на приведенном ранее на рис. 10.

При прохождении сигнала через канал связи, обладающий частотной дисперсией, т. е. вносящий разную групповую задержку на разных частотах, символы оказываются размазанными во времени и наползают друг на друга. В этом случае устранить межсимвольную интерференцию полностью не удается. Чтобы минимизировать ее, используют так называемые адаптивные фильтры, параметры которых подстраиваются под характеристики обрабатываемого сигнала. Такие фильтры очень активно изучаются в настоящее время.

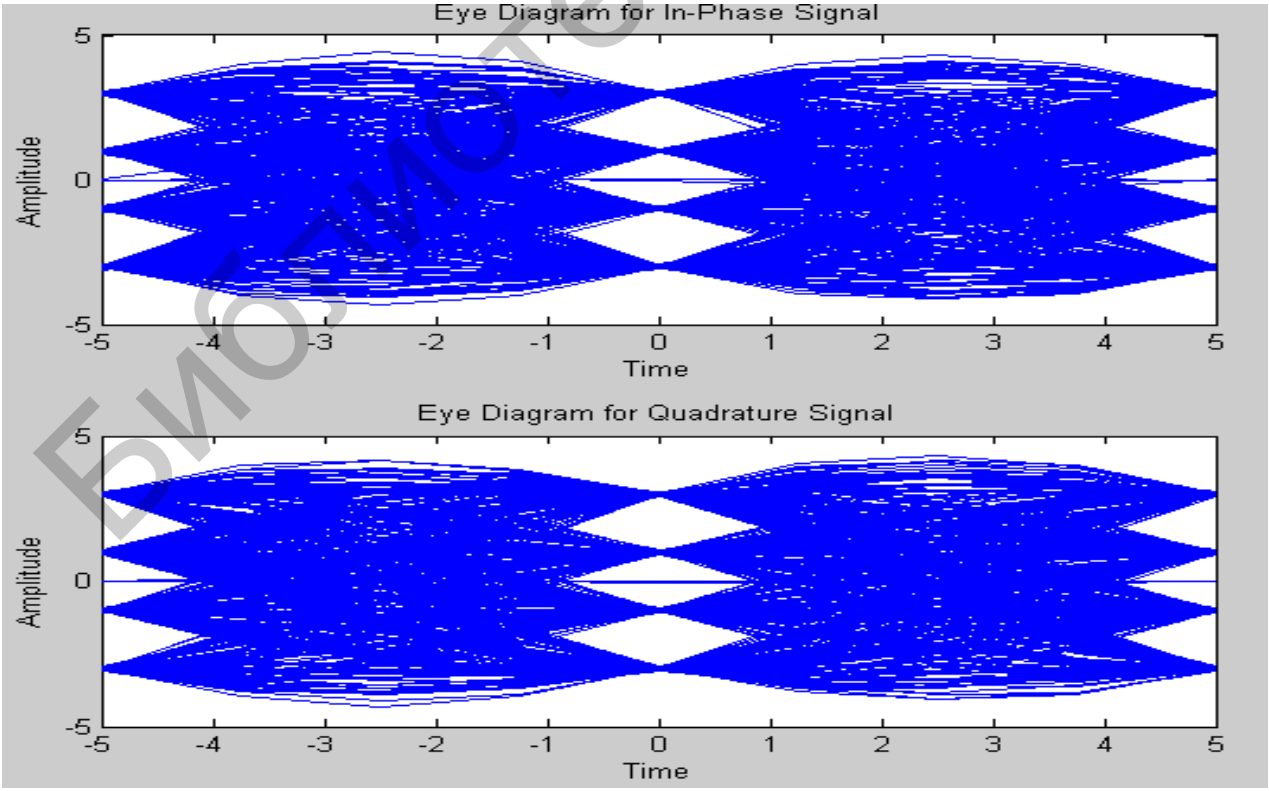


Рис. 14. Глазковая диаграмма,
полученная при приеме сигнала с квадратурной манипуляцией

Задание 5. Выполните моделирование глазковой программы при различных значениях ее параметров. Зарисуйте качественно полученные в результате работы программы и объясните получившиеся при этом различия.

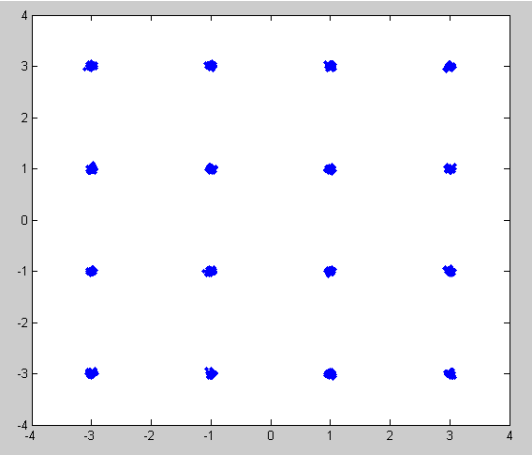


Рис. 15 Диаграмма рассеяния, полученная при приеме сигнала с квадратурной манипуляцией

Пример 6. Создать случайное цифровое сообщение размером 5000 символов и использовать его для квадратурной амплитудной манипуляции 16-позиционного созвездия, передать полученный сигнал через канал связи, зашумленный белым гауссовским шумом, построить диаграмму рассеяния с учетом случайных шумовых данных, демодулировать полученный сигнал на выходе канала связи, чтобы восстановить исходное сообщение, оценить скорость символьной ошибки, проверив выход и построить диаграмму рассеяния.

Выполним следующие коды MATLAB:

```
M=16;%размер созвездия
x=randint(5000,1,M);
y=modulate(modem.qammod(M),x);% используем 16QAM модуляцию, чтобы создать %модулированный сигнал
ynoisy=awgn(y,15,'measured'); % передадим сигнал через канал связи с белым гауссовским шумом
scatterplot(ynoisy);% Получим диаграмму рассеяния, показанную на рис. 16
z=demodulate(modem.qamdemod(M),ynoisy); %демодулируем сигнал ynoisy, чтобы %восстановить исходное сообщение
[num,rt]=symerr(x,z);
num
rt
```

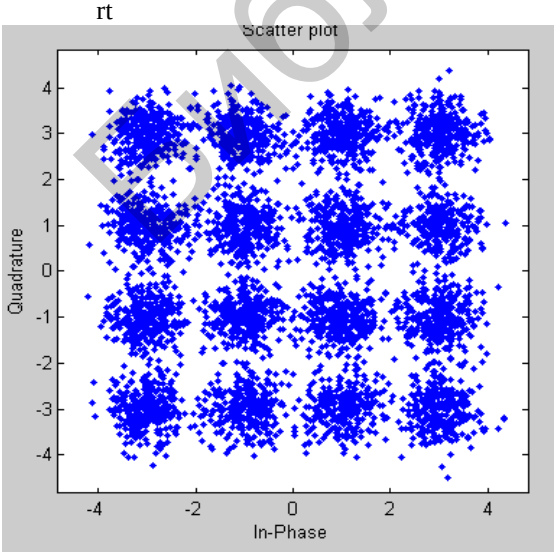


Рис. 16. Диаграмма рассеяния, соответствующая созвездию 16QAM

.

Поясним смысл выполненных процедур при выполнении данной задачи.

Выполнение начнем с генерирования случайного бинарного потока битов. Удобным форматом представления сигнала в MATLAB является вектор или матрица. В этом примере использована функция randint, чтобы создать вектор-столбец, который содержит список последовательных значений бинарного потока битов. Длина этого потока была установлена равной 5000 битов, но это множество может состоять из произвольного числа элементов. Выберем это число равным 30000.

Время дискретизации, связанное с битами, в этом случае не задано явно, и MATLAB не имеет внутренней информации о времени. Для целей данного примера достаточно знания только значений, представленных в потоке битов.

Коды, приведенные ниже, создают stem-диаграмму части битового потока данных. Заметим, что использование оператора colon(:) в MATLAB соответствует выбору некоторой части вектора. За более подробной информацией обратитесь к Colon Operator в документации MATLAB.

Итак, определим параметры создаваемого сигнала.

```
M = 16; % размер сигнального созвездия
k = log2(M); % число битов, приходящихся на один символ
n = 3e4; % длина потока обрабатываемых битов
nsamp = 1; % скорость передискретизации
%устнавливается равной скорости Найквиста

% создаем бинарный поток данных в виде вектор-столбца
x = randint(n,1); % поток случайных бинарных данных

% строим изображения первых 40 битов на stem-диаграмме
stem(x(1:40),'filled');% изобразим 40 первых битов последовательности на %диаграмме, показанной на рис. 17

title('Random Bits – случайные биты');
xlabel('Bit Index – номер бита');
ylabel('Binary Value – бинарное значение бита');
```

Перейдем к процессу подготовки выполнения модуляции. Объект modem.qammod использует M-арный QAM-модулятор. В нашем примере M = 16. Нужно сконфигурировать систему, способную принимать числа, расположенные между 0 и 15 и представленные 4 последовательными битами. Поэтому осуществим подготовку потока данных x перед использованием метода modulate объекта modem.qammod. В частности, нужно получить из каждого 4-битового значения x строки матрицы, используя функцию reshape, и затем применить функцию bi2de, чтобы выполнить преобразование каждого 4-битового числа в соответствующее целое число, представленное в десятичной записи За более подробной информацией об этом отсылаем читателя к документации MATLAB о Reshaping a Matrix.

Первые 40 битов сгенирированной последовательности показаны на рис. 17, а первые 10 символов информационного потока – на рис. 18.

```
% Выполним преобразование битов к символам
% Преобразуем последовательность битов в k-битовые символы
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb');
```

```
% построим диаграмму, показывающую 10 первых символов в виде stem-%диаграммы, показанной на
рис. 18.
%Строим stem-диаграмму, для того чтобы на ней показать первые 10 символов
figure; % Создаем новое окно, в котором разместим первые 10 символов,
%которые формируются случайным образом.
%Обращаем внимание, что при выполнении лабораторной работы может
%получиться stem-диаграмма, которая будет отличаться от приведенной на %рис. 18
stem(xsym(1:10));
title('Random Symbols – случайные символы');
xlabel('Symbol Index – номер символа');
ylabel('Integer Value – численное значение символа');
```

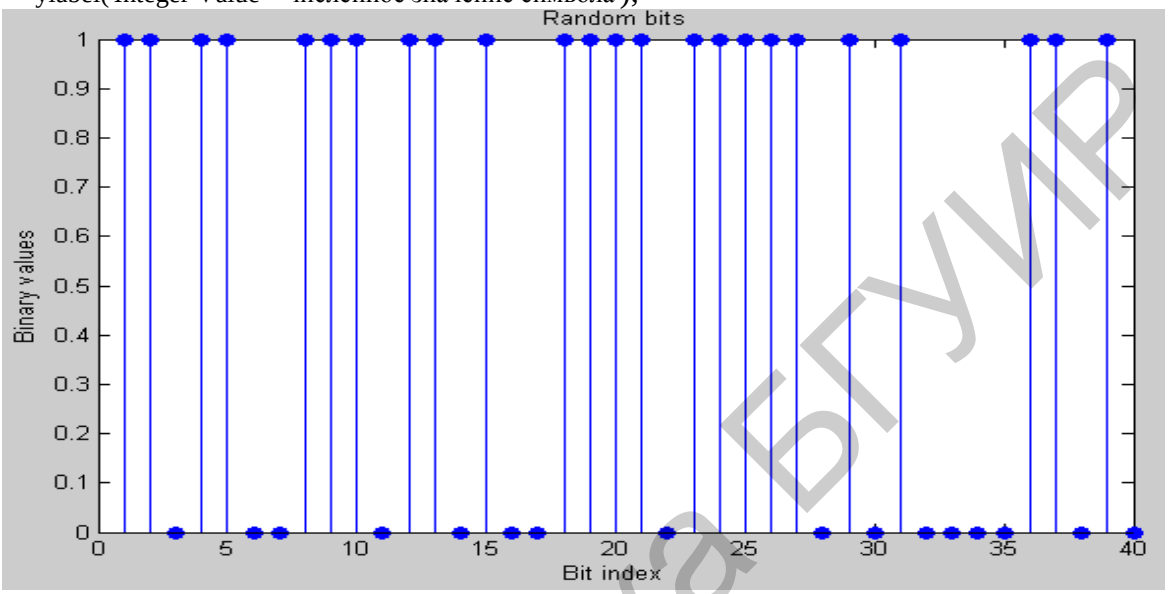


Рис. 17. Первые 40 битов сгенерированной последовательности

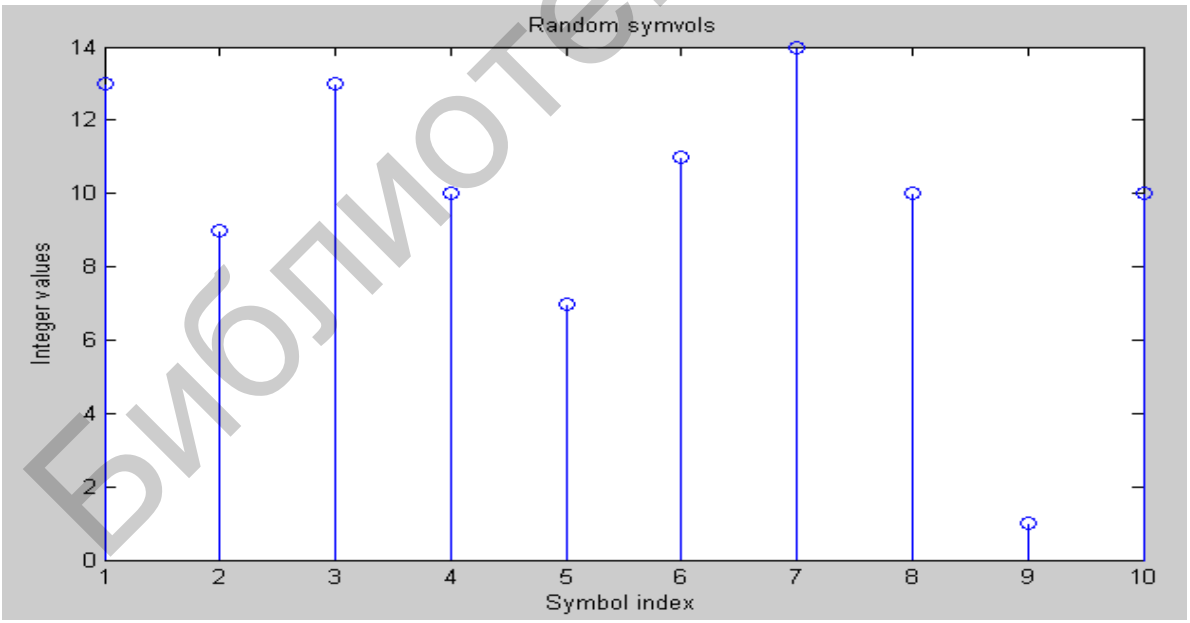


Рис. 18. Первые 10 символов информационного потока

Манипуляция с использованием созвездия сигнала с 16QAM

Имея определенный xsym как вектор-столбец, содержащий целые числа, расположенные между 0 и 15, можно использовать метод modulate объекта modem.qammod, чтобы осуществить модуляцию сигналом xsym. Напомним, что

M = 16 – это размер алфавита, с помощью которого осуществляется передача потока данных. Итак, выполним модуляцию.

```
y=modulate(modem.qammod(M),xsym);
```

Результат получается в виде вектор-столбца, состоящего из комплексных чисел, которые располагаются в 16-ти точках созвездия сигнала с QAM. Чтобы разобраться с этими понятиями лучше, следует обратиться к разделу Modulation документации MATLAB. Заметим, что метод modulate объекта modem.qammod не применяется к импульсам произвольной формы. Об этом можно прочитать, обратившись к Pulse ShapingUsing a Raised Cosine Filter документации MATLAB.

Аддитивный белый гауссовский шум

Применение функции awgn к модулированному сигналу добавляет к нему этот шум. Отношение энергии сигнала к спектральной плотности мощности шума E_b/N_0 вообще является произвольным. Но в данном случае устанавливается на уровне 10 дБ. Выражение, что преобразует это значение в соответствующую величину отношения SNR, включает k, т. е. число битов, приходящихся на один символ (в нашем случае $k = 4$ для сигнала с 16QAM), и nsamp, равный коэффициенту передискретизации, который в нашем случае равен 1. Коэффициент k используется, чтобы преобразовать E_b/N_0 в эквивалентное ему отношение E_s/N_0, которое представляет собой отношение энергии символа к спектральной плотности мощности шума. Коэффициент nsamp используется для преобразования E_b/N_0 в отношение символьной скорости полосы к SNR в полосе дискретизации. Заметим, что определения терминов ytx, yrx, nsamp в определении snr не имеют большого значения в данном примере, однако помогут быстрее разобраться в более сложных примерах использования импульсов разной формы.

Итак, переданный сигнал имеет вид

```
ytx=y;
% Передача по каналу с аддитивным гауссовским белым шумом определяется
%командами
EbN0=10; % в дБ
snr=EbN0 + 10*log10(k) – 10*log10(nsamp);
ynoisy=awgn(ytx,snr,’measured’);
%Принятый сигнал приобретает форму
yrx=ynoisy;
```

Чтобы получить больше сведений о функции awgn и других функциях канала связи, следует обратиться к Channels документации MATLAB.

Создание диаграммы рассеяния

Применение функции рассеяния к переданному и принятому сигналам показывает, что созвездие принятого сигнала выглядит как созвездие переданного сигнала, но искажено шумом. На диаграмме горизонтальная ось соответствует синфазной компоненте сигнала и вертикальная ось – квадратурной компоненте. Приведенные ниже коды используют заголовок и функции для названий осей, принятых в системе MATLAB.

```
%% Scatter Plot
% Create scatter plot of noisy signal and transmitted
% signal on the same axes.
h = scatterplot(yrx(1:nsamp*5e3),nsamp,0,'g.');
hold on;
scatterplot(ytx(1:5e3),1,0,'k*',h); % создаем диаграмму рассеяния и
```

```
%покажем переданный сигнал на тех же осях

title('Received Signal – принятый сигнал');
legend('Received Signal– принятый сигнал ','Signal Constellation – сигнальное созвездие');
axis([-5 5 -5 5]); % установим пределы изменения переменных

hold off;
```

Диаграмма рассеяния вместе с созвездием сигнала 16QAM показана на рис. 19.

Для получения дополнительных сведений о функции scatterplot можно обратиться к Scatter Plots в документации MATLAB.

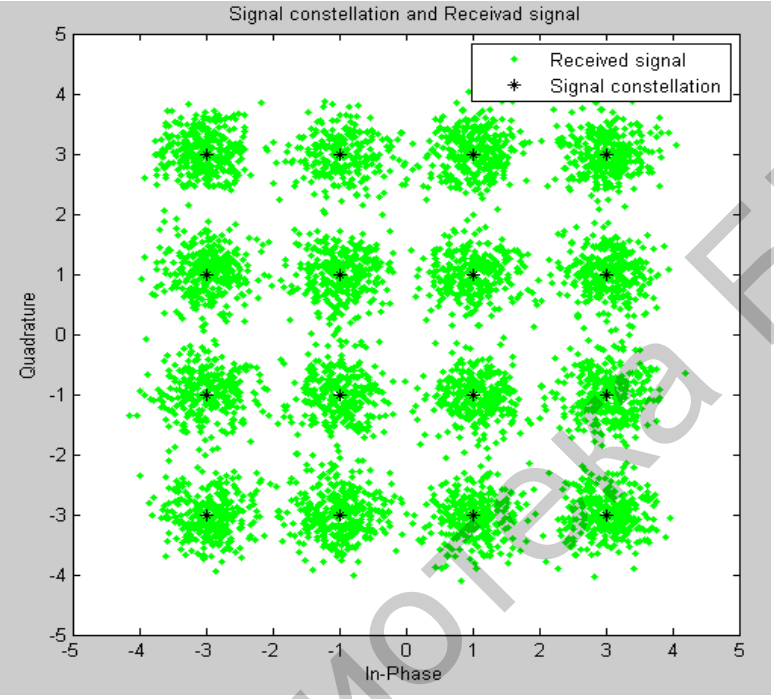


Рис. 19. Диаграмма рассеяния и созвездие сигнала с 16QAM

.

Деманипуляция с использованием сигнала с 16QAM

Применяя метод demodulate объекта modem.qamdemod к принятому сигналу, деманипулируем его. Результат получается в виде вектора, содержащего целые числа, расположенные между 0 и 15. Имеем

zsym=demodulate(modem.qamdemod(M),yrx);

Преобразование целых значений сигнала в бинарное представление

На предыдущих этапах решения поставленной задачи был создан сигнал zsym, представляющий собой вектор целых чисел. Чтобы получить эквивалентный ему бинарный сигнал, используем функцию de2bi, с помощью которой преобразуем каждое целое число в соответствующую бинарную четырехбитовую последовательность, записанную в виде строки матрицы.

Затем, используя функцию reshape, соберем все биты в один вектор-столбец, заменив им четырехстолбцовую матрицу.

Итак, выполним преобразование символьной формы принятого сигнала к битовой форме. (Раньше было осуществлено преобразование битовой формы сигнала к его символьной форме). Имеем

```
z=de2bi(zsym, 'left-msb'); % Преобразование чисел в битовую форму
z=reshape(z.',prod(size(z),1)); % Преобразование матрицы в вектор
```

Изображение созвездий QAM-сигнала

Мы научились создавать диаграммы рассеяния по результатам визуализации значений QAM-сигнала. Эти диаграммы показывают точки созвездия QAM-сигнала, но информации о том, каким числам между 0 и 15 они соответствуют, не дают. Тем не менее созвездие, характеризующее объект modem.qammod, содержит все точки созвездия сигнала с 16-QAM. Построим это созвездие.

```
M = 16; % число точек в созвездии
h = modem.qammod(M); % объект модулятора
mapping = h.SymbolMapping; % вектор, изображающий символы
pt = h.Constellation; % вектор всех точек в созвездии

% построение созвездия
scatterplot(pt);
```

Изображение созвездия сигнала с QAM показано на рис. 20.

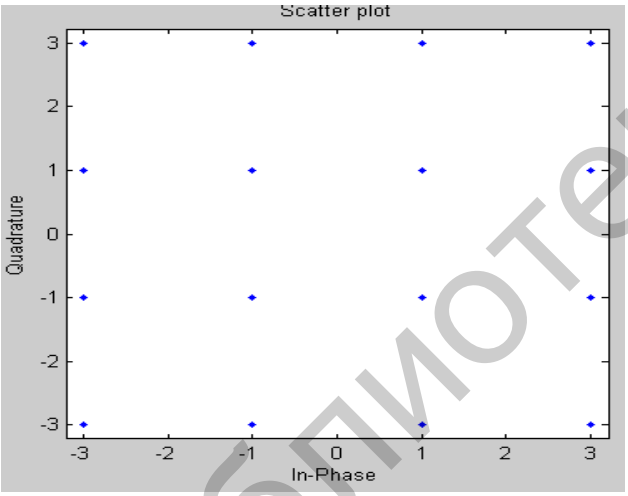


Рис. 20. Созвездие сигнала с 16-QAM

Используем функцию text, чтобы разместить числа на диаграмме рядом с соответствующей точкой созвездия. Координаты размещения этих чисел находятся вблизи точек созвездия, задаваемых их действительными и мнимыми составляющими, но мы их немного сдвинем, чтобы избежать перекрытия. Текст изображения чисел представим в бинарном виде. (Заметим, что функция dec2bin в MATLAB создает строку цифровых символов, в то время как функция de2bi используется для создания вектора чисел).

```
% покажем числа рядом с точками созвездия
text(real(pt)+0.1,imag(pt),dec2bin(mapping));
axis([-4 4 -4 4]); % изменим масштаб на осях, чтобы можно было
%разместить числа на этой диаграмме
```

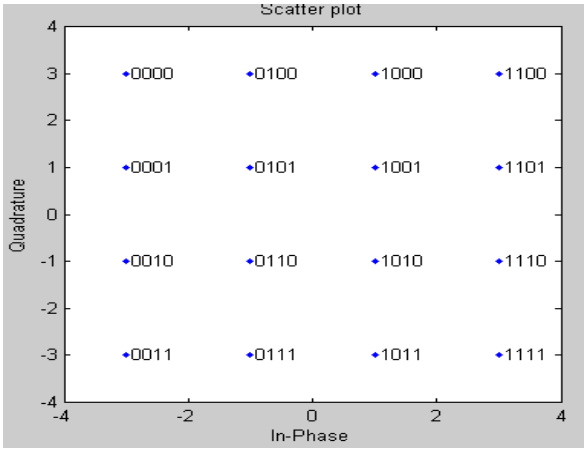


Рис. 21. Созвездие
сигнала с 16-QAM
с числами, соответствующими
значениям сигналов

На диаграмме, приведенной на рис. 21, точки 0001 и 0010 соответствуют соседним точкам, находящимся на левой стороне созвездия. Поскольку эти бинарные представления отличаются двумя битами, смежность их указывает, что объект modem.qammod не использует созвездие сигнала, кодированного в соответствии с кодом Грэя. Иными словами, если бы соседние точки отличались только одним битом, то тогда бы они кодировались по коду Грэя. Поэтому сформируем созвездие в соответствии с требованиями кода Грэя.

```
%% модифицируем предшествующую диаграмму созвездия
M = 16; % число точек в созвездии
h = modem.qammod('M',M,'SymbolOrder','Gray'); % объект модулятора
mapping = h.SymbolMapping; % вектор, изображающий символы
pt = h.Constellation; % вектор всех точек в созвездии

scatterplot(pt); % изображаем созвездие

% модифицируем диаграмму в соответствии с требованиями кода Грэя
text(real(pt)+0.1,imag(pt),dec2bin(mapping));
axis([-4 4 -4 4]); % изменим масштаб на осях, чтобы можно было
%разместить числа на этой диаграмме
```

Созвездие сигнала с 16-QAM с числами, соответствующими значениям сигнала, кодированного по коду Грэя, показано на рис. 22.

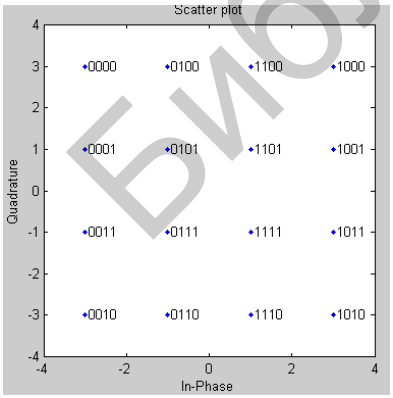


Рис. 22. Созвездие
сигнала с 16-QAM
с числами, соответствующими
значениям сигнала,
кодированного
по коду Грэя

Задание 6. Выполните 2 – 3 раза последовательность программ, приведенных в примере 6, изменяя значения вводимых параметров, чтобы хорошо понять смысл выполненной работы.

3.3. Моделирование дифференциальной фазовой манипуляции

Бинарная фазовая манипуляция (Binary Phase Shift Keying, BPSK) является достаточно сложным процессом, изучение которого осуществляется во многих учебных планах университетских курсов, в частности, в дисциплинах, связанных со спутниковыми навигационными системами, которые формируют в околоземном пространстве радионавигационное поле. Средства, обеспечивающие навигацию, т. е. спутниковые навигационные приемники, станции с дифференциальным режимом (контрольно-корректирующие станции), аппаратура обслуживания и ряд других радиотехнических устройств принимают и обрабатывают информацию из радионавигационного поля и решают задачи в соответствии со своим функциональным назначением. Сигналы навигационных спутников, как на спутниках, так и в навигационных приемниках потребителя, подвергаются специальной обработке для эффективной передачи, поиска, обнаружения, слежения, измерения в условиях помех данных без потери информации.

Здесь рассматривается один из важных компонентов системы спутниковой радионавигации – метод дифференциальной бинарной фазовой манипуляции, (Differential Binary Phase Shift Keying, DBPSK), используемый в системе ГЛОНАСС. Этот метод реализуется посредством перекодировки исходной последовательности информационных символов по следующему алгоритму:

$$a_{вых\ i} = a_{вх\ i} \oplus a_{вых\ i-1}, \quad (5)$$

где $a_{вх\ I}$, $a_{вых\ I}$ – входная и выходная последовательности символов при передаче соответственно, а знак ○ соответствует операции сложения по модулю два.

При приеме перекодировка выполняется по принципу:

$$b_i = a_{вых\ i} \oplus a_{вых\ i-1}, \quad (6)$$

где b_i – последовательность символов после перекодировки на выходе приемника.

Рассмотрим процесс получения колебаний с дифференциальной фазовой манипуляцией, использующий графические возможности систем SIMULINK и MATLAB.

Идея дифференциального кодирования состоит в том, что передается не абсолютное значение информационного символа, а его изменение (или не изменение) относительно предыдущего значения, т. е. каждый последующий передаваемый символ содержит в себе информацию о предыдущем символе. Тем самым для извлечения исходной информации в качестве опорного сигнала можно использовать не несущую частоту, а предыдущее значение символа. В самом деле, если в приемнике осуществить задержку принятого символа на один символьный интервал, а затем произвести перемножение полученного и задержанного символов, то результатом этой операции будет исходная информационная последовательность. После фильтрации с помощью ФНЧ или согласованного фильтра остается только постоянная составляющая. Очевидно, что ни временная форма, ни спектральный состав DBPSK сигнала не будет отличаться от обычного BPSK сигнала.

Из блоков библиотеки SIMULINK соберите схему, представленную на рис. 23. В этой схеме используются следующие блоки: блок Repeating Sequence

Interpolated содержит данные (символы), которые надлежит перекодировать, чтобы получить последовательность символов с дифференциальной фазовой манипуляцией. Блоки Unit Delay выполняют задержку на один символ, блоки Logical Operator XOR – это сумматоры по модулю два. Блок Scope – представляет собой трехканальный осциллограф, регистрирующий результаты работы схемы.

Символы [11010100100] в сигнальной форме представлены на верхнем графике рис. 24 и соответствуют данным, передаваемым по соответствующему каналу на рис. 23. Средний график рис. 24 – это результат выполнения алгоритма (5) и соответствует данным, сформированным в другом канале рис. 23. Нижний график рис. 24 представляет собой результат применения алгоритма (6) восстановления данных и соответствует данным, сформированным в соответствующем канале на рис. 23. Из сопоставления графиков рис. 24 видно, что на нижний вход осциллографа подается сигнал, получающийся в результате обратного преобразования сигнала с фазовой манипуляцией в исходный сигнал, поступающий на верхний вход осциллографа.

Моделирование преобразований по алгоритмам (5) и (6) можно выполнить и непосредственно в системе MATLAB, для чего необходимо создать m-файл, с помощью которого будет реализована процедура перекодировки по алгоритмам (5) и (6):

```
%m-файл процедуры перекодировки
%входная последовательность
a = [11010100100];
%выходная последовательность
aout = a(1);
i = 1;
for i= 5
          aout (i) = xor(a(i), aout(i-1));
end
          aout
%вывод выходной последовательности в командное окно
b(1) = aout(1);
for I = 2:5
          b(i) = xor(aout(i-1), aout(i))
end
b
%вывод обратного преобразования последовательности в командное окно
```

Выполняя этот m-файл, легко убедиться, что результат выполнения соответствует данным, изображенным в командном окне.

Задание 7. Выполните 2 – 3 раза последовательность программ, приведенных в описанных примерах, изменяя значения вводимых параметров, чтобы хорошо понять смысл выполненной работы.

Изложенный метод можно применять и к другим видам фазовой манипуляции, например, к дифференциальной фазовой манипуляции со смещением на π/4 (π/4 Differential Quadrate Phase Shift Keying, π/4DQPSK), которая является формой дифференциальной фазовой манипуляции, специально разработанной для четырехуровневых квадратурно-манипулированных сигналов (Quadrate Phase Shift Keying, QPSK). Сигнал этого вида модуляции может быть демодулирован некогерентным детектором, как это свойственно сигналам DBPSK

модуляции. Отличие дифференциального кодирования в π/4DQPSK модуляции от дифференциального кодирования в DBPSK модуляции состоит в том, что передается не относительное изменение модулирующего цифрового символа, а относительное изменение модулируемого параметра, в данном случае фазы. Подробный анализ показывает, что π/4DQPSK сигнал лучше, чем DBPSK или QPSK сигналы, сохраняется при многолучевом распространении и федингe, характерных для мобильной связи.

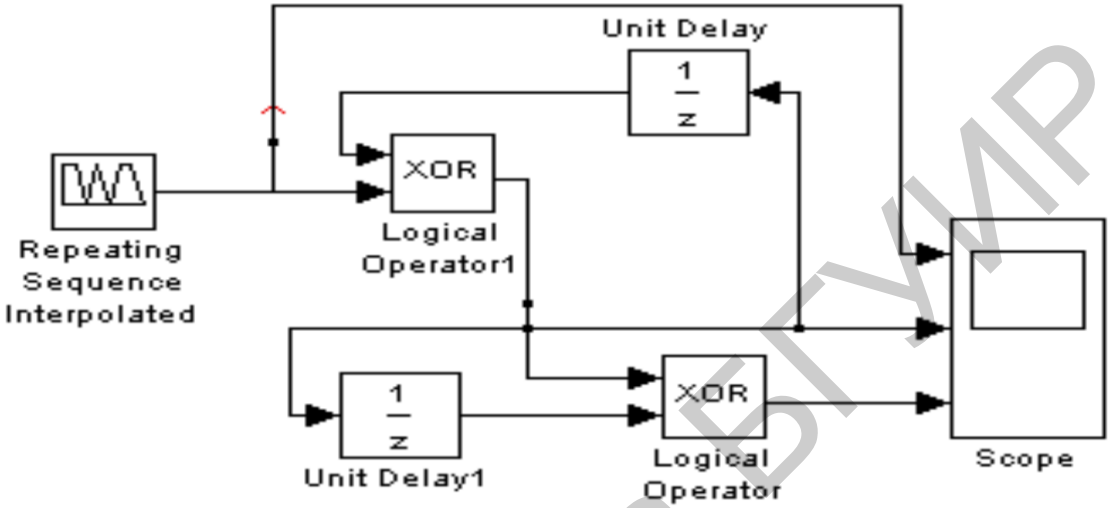


Рис. 23. Модель формирования относительной фазовой манипуляции, построенная в системе SIMULINK

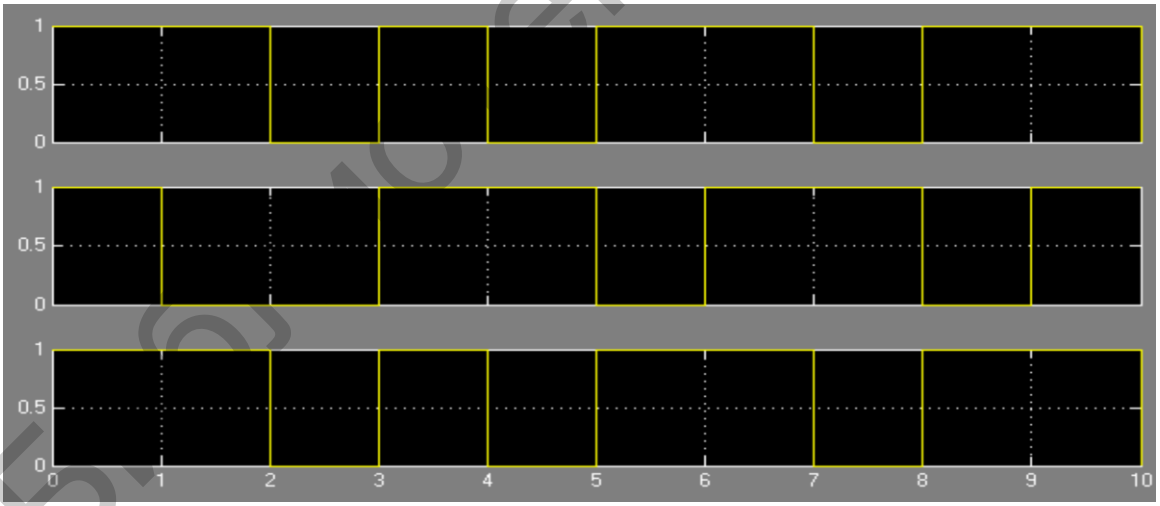


Рис. 24. Осциллограммы исходного сигнала с фазовой манипуляцией (верхний рисунок), сигнала с относительной фазовой манипуляцией (средний рисунок) и обратно преобразованного сигнала с относительной фазовой манипуляцией в сигнал с фазовой манипуляцией (нижний рисунок)

Перейдем к более подробному описанию выполнения лабораторной работы.

1. Войдем в программу **Simulink.**

2. Выберем последовательно следующие опции **File –> New –> Model** в **Simulink Library Browser**, чтобы сформировать новую модель.

3. Выберем опции **Communications Blockset** -> **Comm Sources** -> **Random Data Sources**, а затем выберем блок **Random Integer Generator** в модельном окне программы. Осуществив двойной щелчок мышью на этом блока, установим следующие его параметры:

- M−ary number to 2
- Initial seed to 37
- Sample time to 0.1

= Output Data Type to double

4. Для иллюстрации выполненных действий приведем рис. 25, на котором представлена папка **Simulink Library Browser**

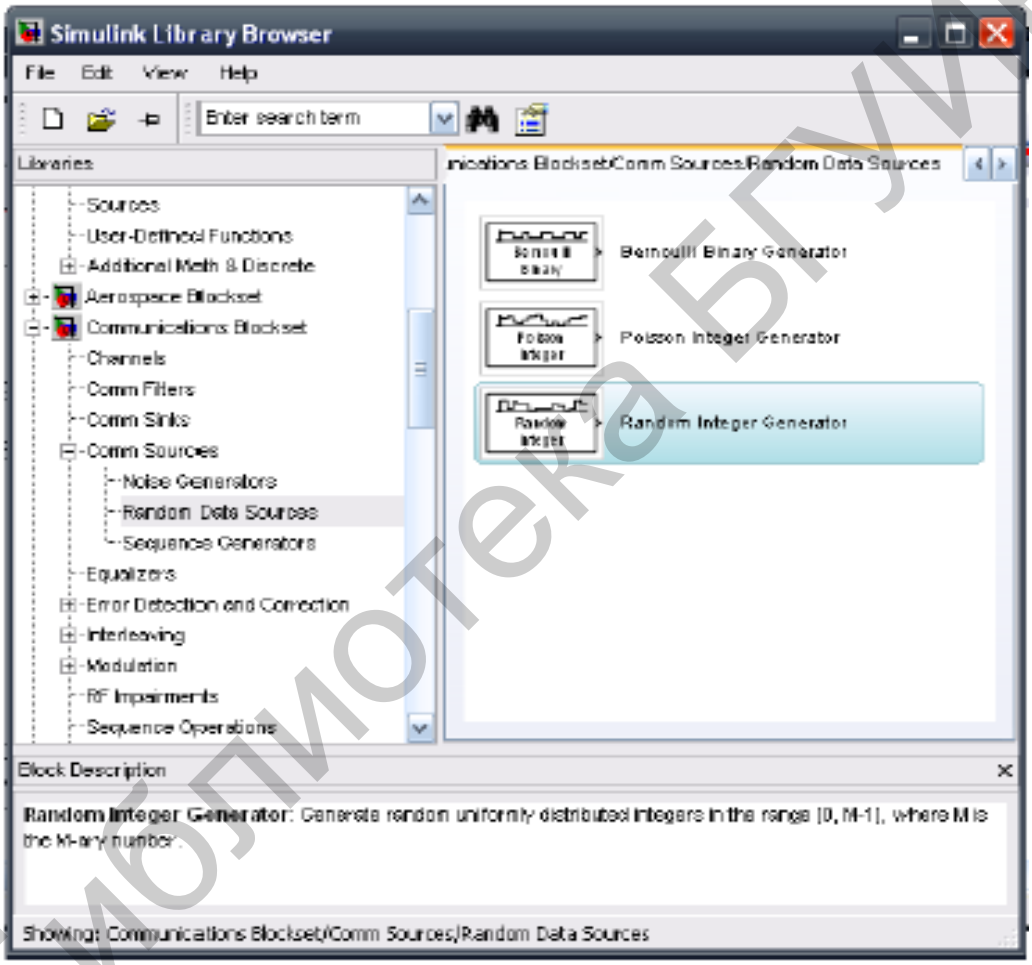


Рис. 25. Общий вид папки **Simulink Library Browser**

5. Выберем опции **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** > **AM**, установим модуль **M-PAM Modulator Baseband** в модельном окне программы. Щелкнув дважды мышью на установленном блоке, установим следующие его параметры:

- M−ary number to 2
- Input type to Integer
- Constellation ordering to Binary

- Normalization method to Peak Power

- Peak power (watts) to 1

- Samples per symbol to 1

6. Выберем опции **Communications Blockset** -> **Channels,** установим модуль **AWGN Channel** в модельном окне программы. Щелкнув дважды мышью на установленном блоке, установим следующие его параметры:

- Initial seed to 37

- Mode to Signal-to-noise ratio (Es/No)

- Es/No (dB) to 10

- Input signal power (watts) to 1

- Symbol period(s) to 0.1

7. Выберем опции **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** > **AM**, установим модуль **M-PAM Demodulator Baseband** в модельном окне программы. Щелкнув дважды мышью на установленном блоке, установим следующие его параметры:

- M−ary number to 2

- Output type to Integer

- Constellation ordering to Binary

- Normalization method to Peak Power

- Peak power (watts): to 1

- Samples per symbol to 1

8. Для иллюстрации выполненных действий приведем рис. 26, на котором представлена папка **Function Block Parameters: PAM Demodulator Baseband.**

9. Выберем опции **Communications Blockset** -> **Comm Sinks**, установим модуль **Error Rate Calculation** в модельном окне программы.

10. Перейдем к **Simulink** -> **Sinks**, установим модуль **Display** в модельном окне программы. Увеличим размеры этого модуля, чтобы разместить на нем все необходимые входы.

11. Перейдем к **Communications Blockset** -> **Comm Sinks** и перетащим два модуля **Discrete-Time Scatter Plot Scope** в модельное окно программы.

12. Перейдем к **Simulink** -> **Math Operations** и установим два модуля **Complex to Real−Imag** в модельном окне программы.

13. Перйдем к **Simulink** -> **Sinks** и установим два модуля **XY Graph** в модельном окне программы.

14. Снова вернемся к **Simulink** -> **Sinks** и установим два модуля **Scope** в модельном окне программы.

15. Перейдем теперь к **Simulink Extras** -> **Additional Sinks** и установим два модуля **Power Spectral Density** в модельном окне программы.

16. Соединим установленные блоки между собой, как показано на рис. 27.

17. Установим следующие параметры моделирования в папке **Simulation** -> **Configuration Parameters**,: общий вид которой показан на рис. 28:

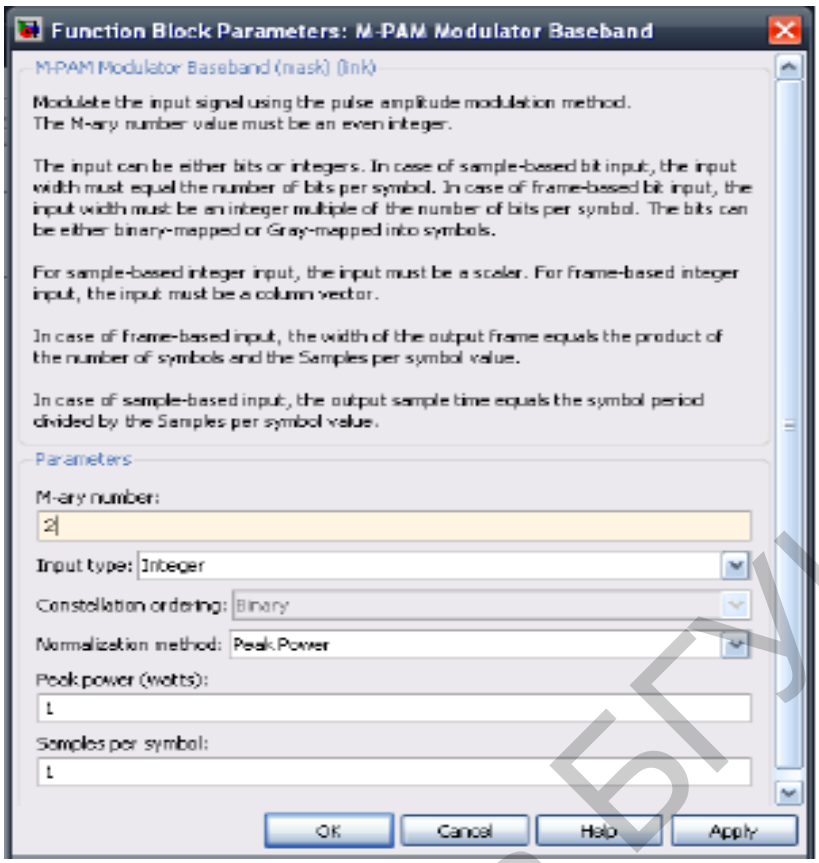


Рис. 26. Общий вид папки **Function Block Parameters: PAM Demodulator Baseband**

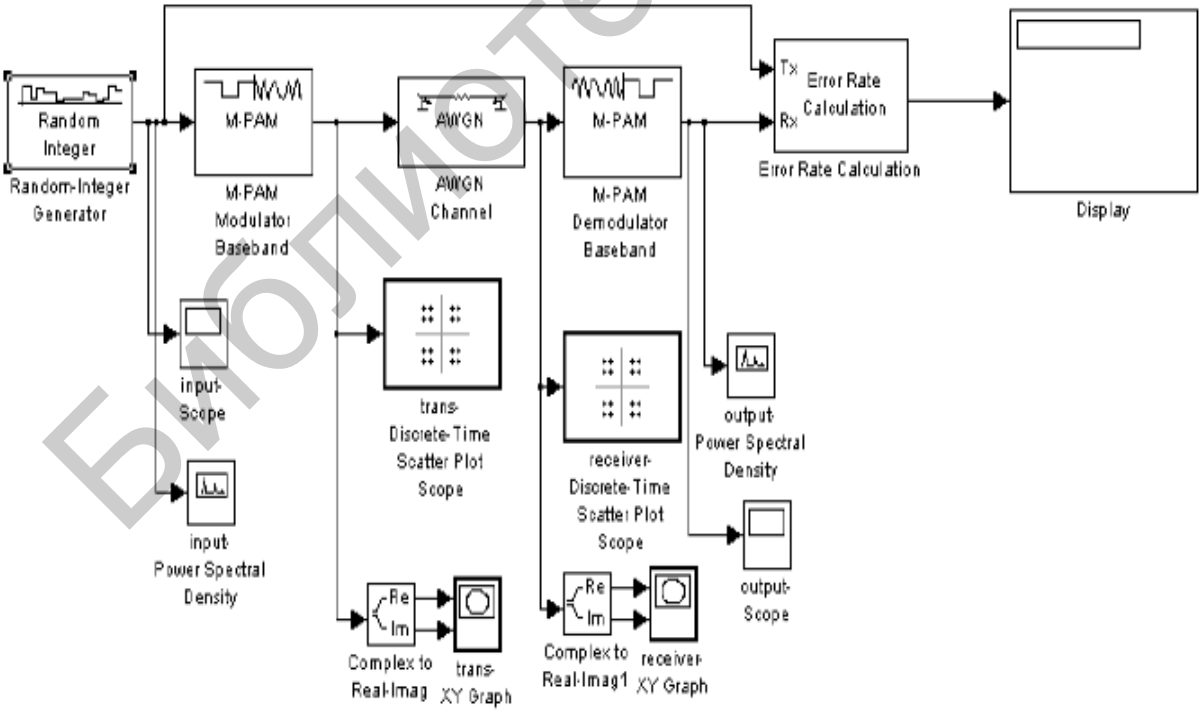


Рис. 27. Структурная схема моделирования амплитудной модуляции

- Start time to 0.0
- Stop time to 100.0
- Type to Variable-step
- Solver to discrete (no continuous states)
- Max. step size to auto

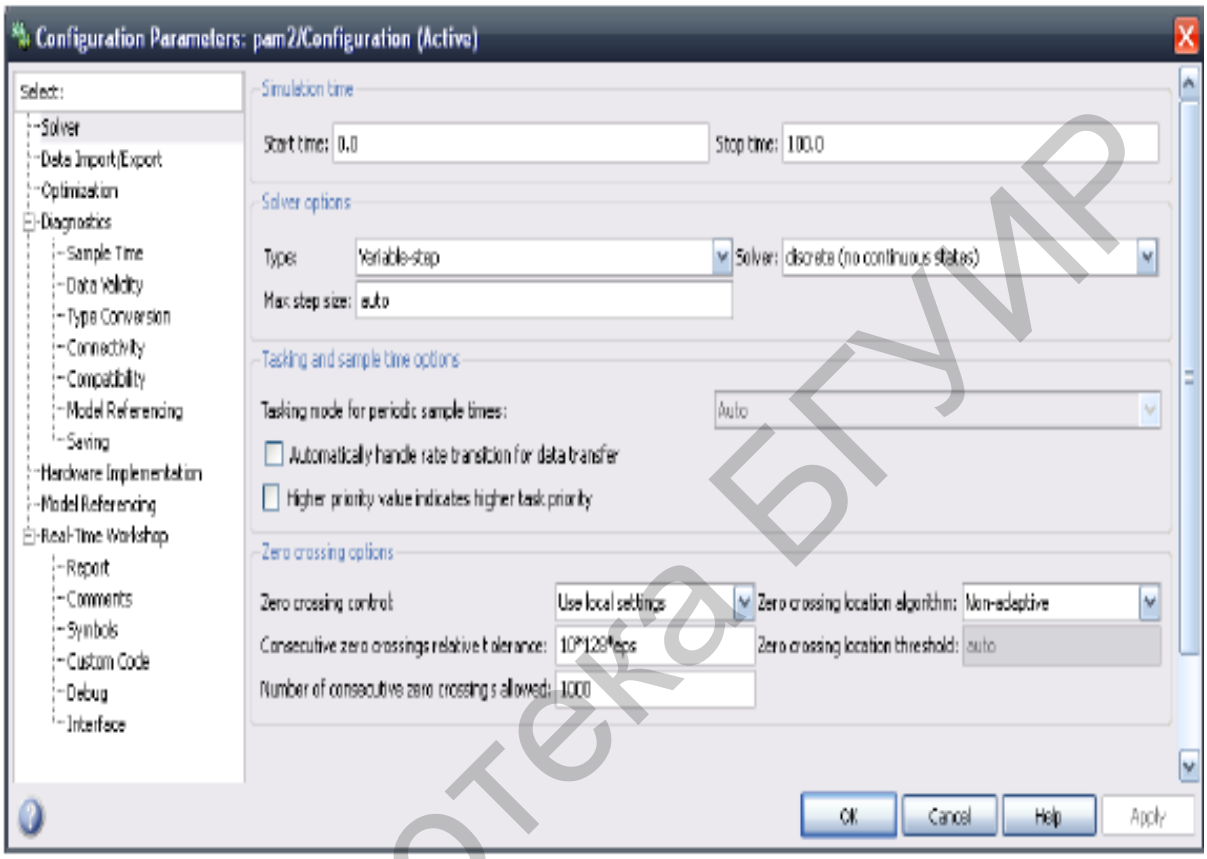


Рис.28. Общий вид папки **Configuration Parameters**

18. Запустим программу на выполнение **Simulation** -> **Start**, наблюдаем результаты работы программы, сохраним все диаграммы и значения в модуле **Display**.

19. Моделируем снова, изменяя М-ичные числа, выбирая их равными 4, 8, 16, 32 и 64 и корректируя параметры модулей **Random Integer Generator**, **M-PAM Modulator Baseband** и **M-PAM Demodulator Baseband**. Параметры модуля **Random Integer Generator** можно установить из папки, приведенной на рис. 29.

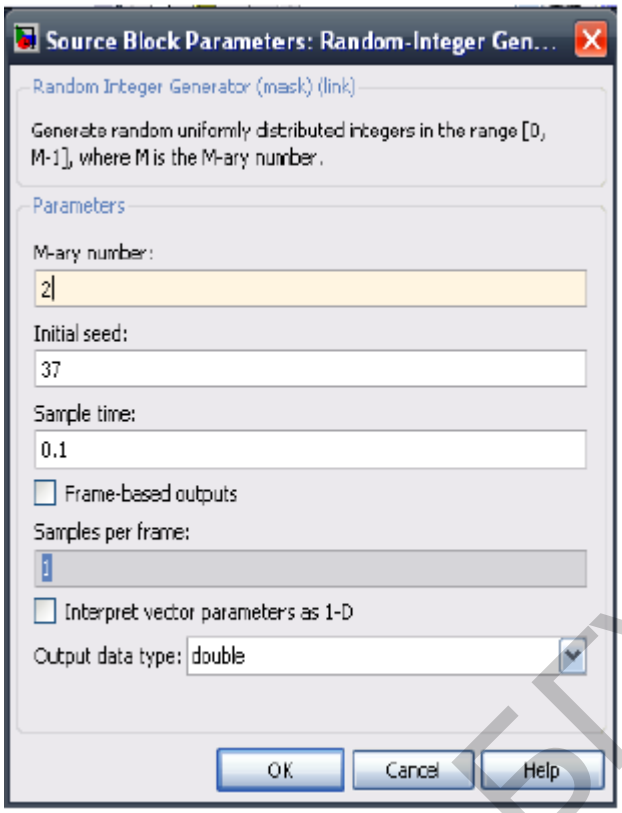


Рис. 29. Папка для выбора параметров модуля **Random Integer Generator**

Моделирование фазовой манипуляции проводится аналогичным образом. Сначала соберем схему лабораторной установки, показной на рис. 30, а затем установим значения используемых в этой схеме параметров. Для этого воспользуемся папками, приведенными на рис. 31…34.

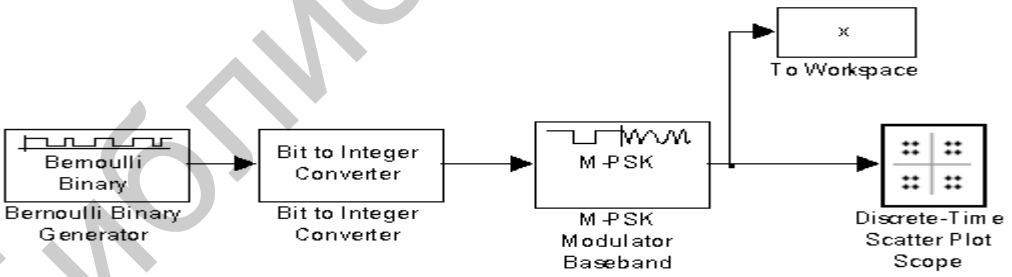


Рис. 30. Моделирование фазовой манипуляции

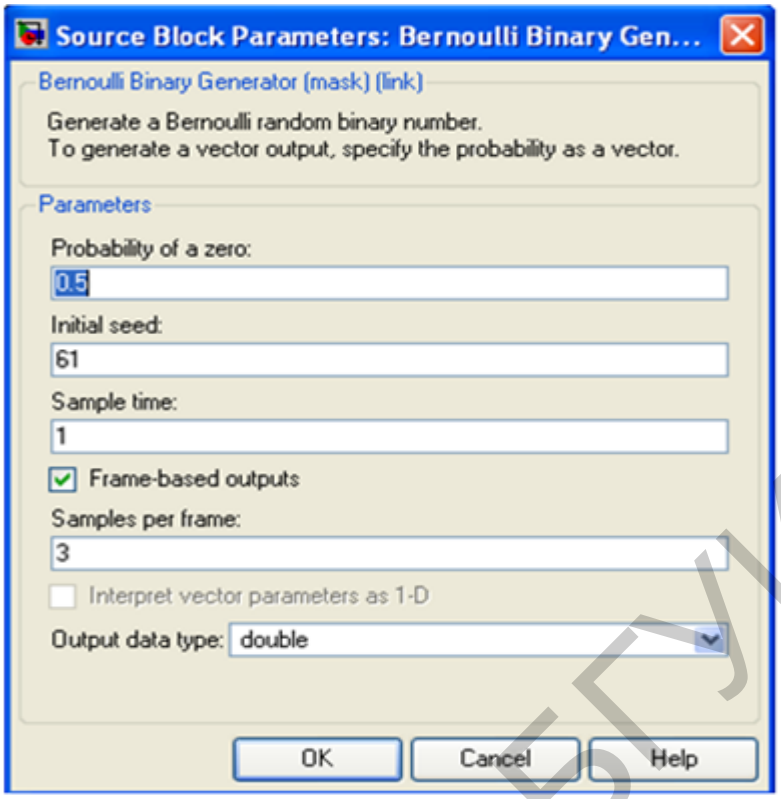


Рис. 31. Папка для установления параметров модуля **Bernoully Binary Generator**

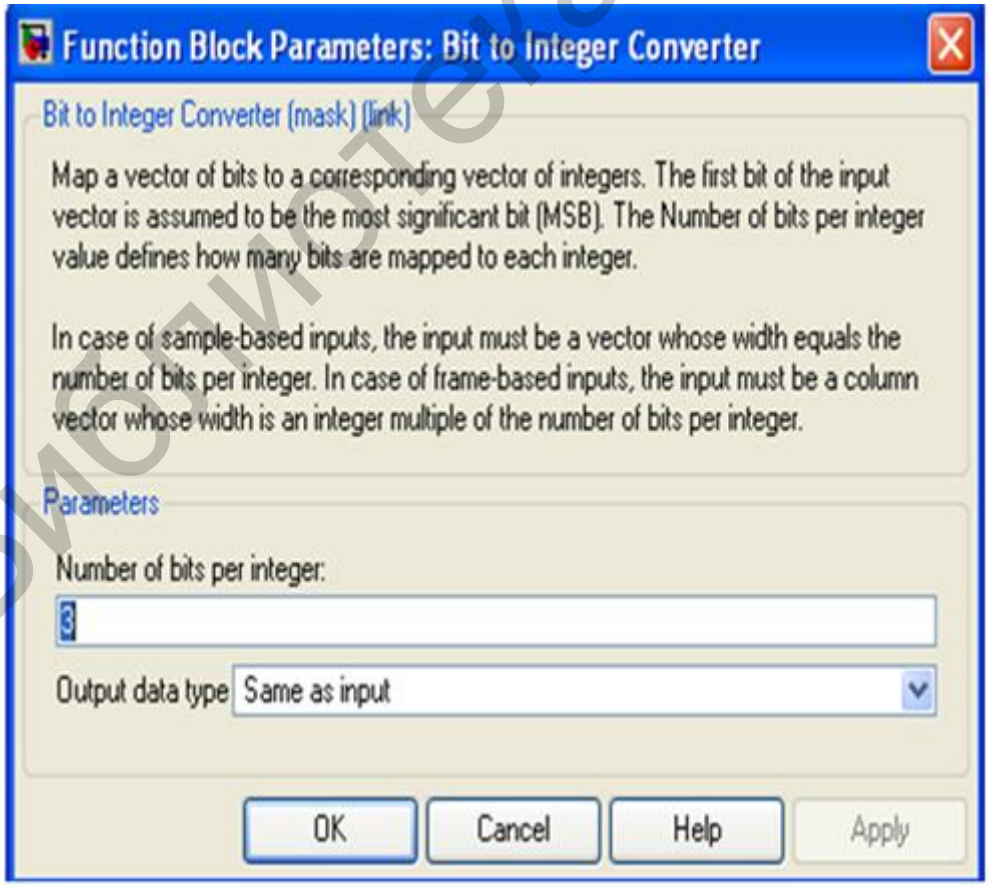


Рис. 32. Папка для установления параметров модуля **Bit to Integer Converter**

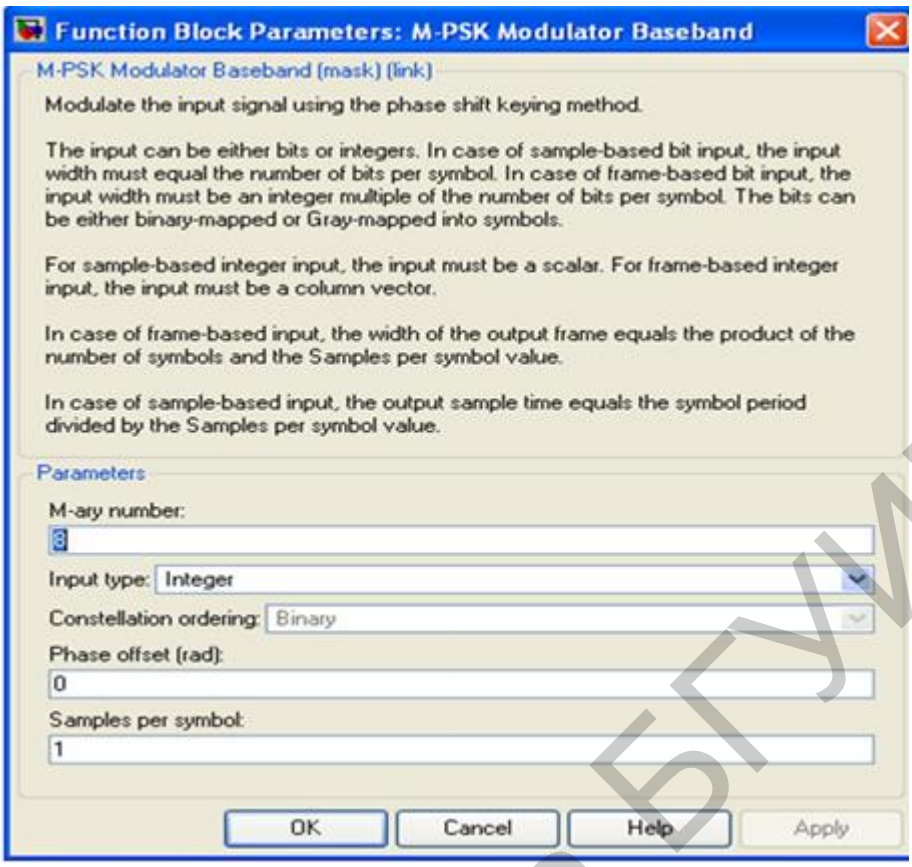


Рис. 33. Папка для установления параметров модуля **M-PSK Modulator Baseband**

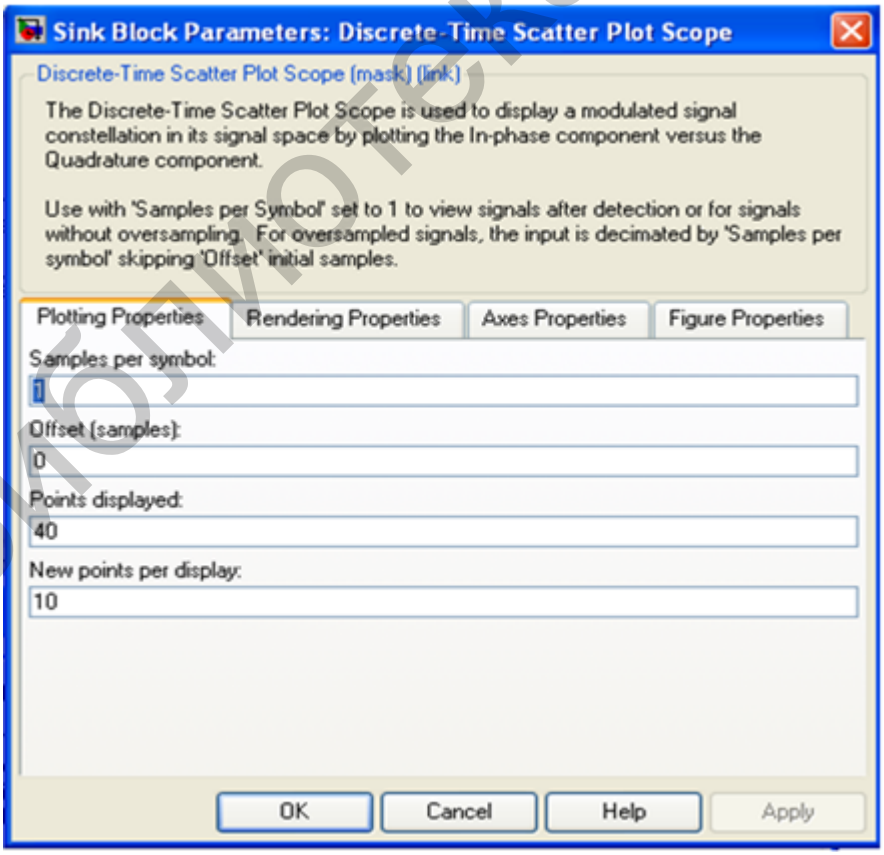


Рис. 34. Папка для установления параметров модуля **Discrete Time Scatter Plot Scope**

Для изучения процессов фазовой манипуляции в системе с каналом с аддитивным гауссовским шумом можно собрать схему, приведенную на рис. 35.

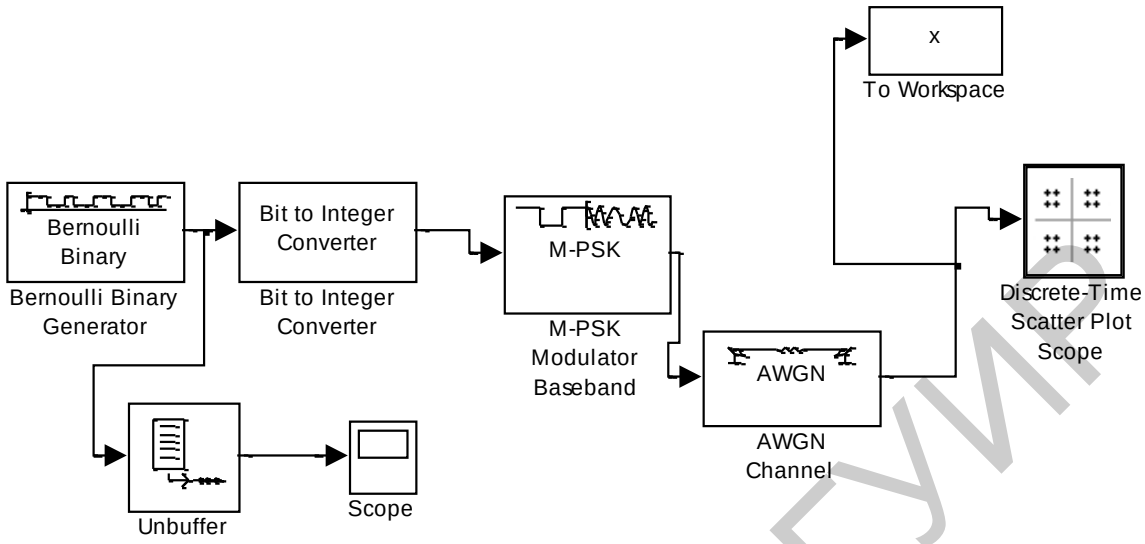


Рис. 35. Моделирование фазовой манипуляции в канале с аддитивным гауссовским шумом

Моделирование квадратурной фазовой манипуляции можно рассмотреть на примере изучения процессов, протекающих в схеме, приведенной на рис. 36.

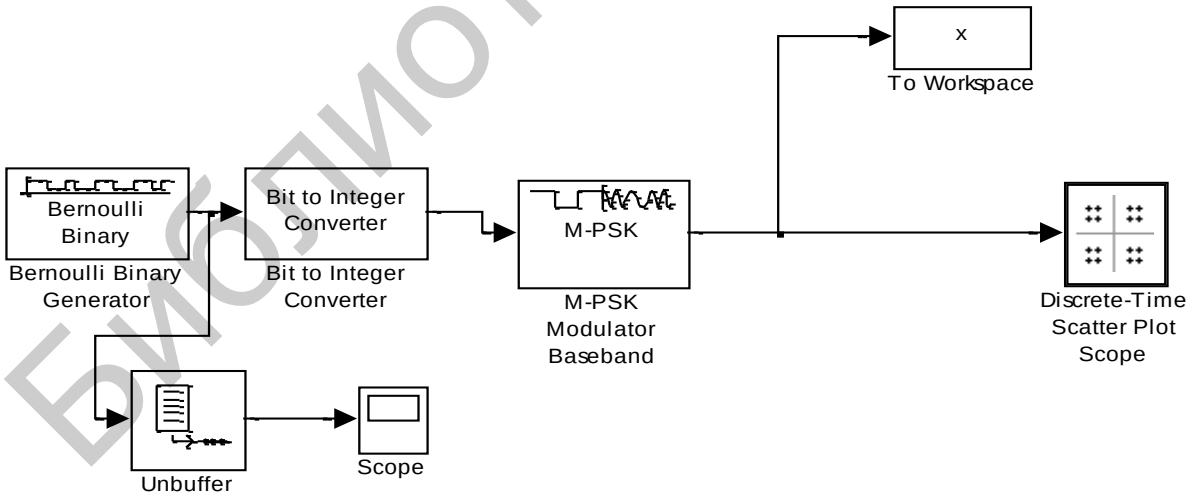


Рис. 36. Моделирование квадратурной фазовой манипуляции

Более полное представление о квадратурной фазовой манипуляции можно получить, проведя исследование процессов, протекающих в схеме, приведенной на рис. 37.

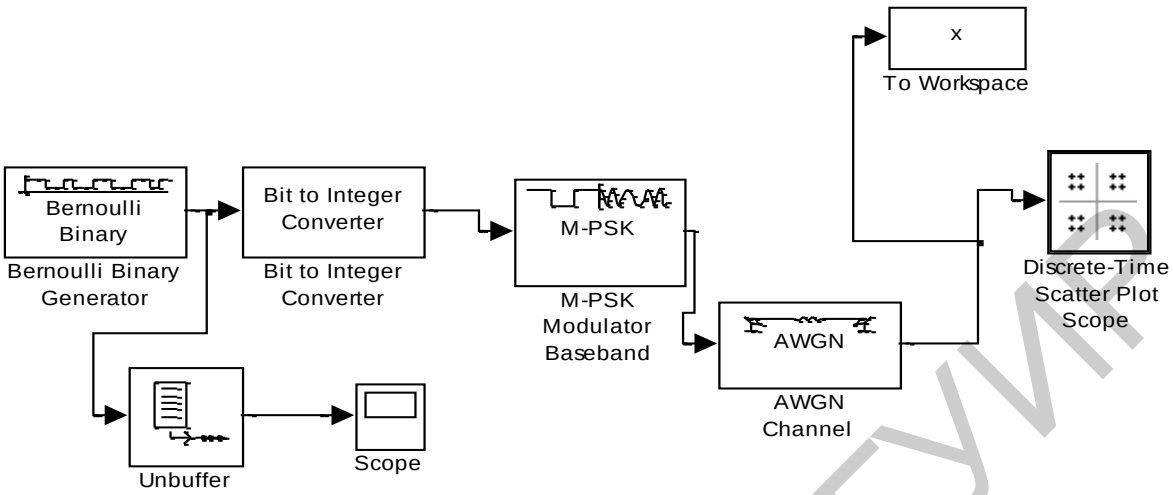


Рис. 37. Моделирование квадратурной фазовой манипуляции в канале с аддитивным гауссовским шумом

Изучению процессов относительной квадратурной фазовой манипуляции может способствовать схема, приведенная на рис. 38.

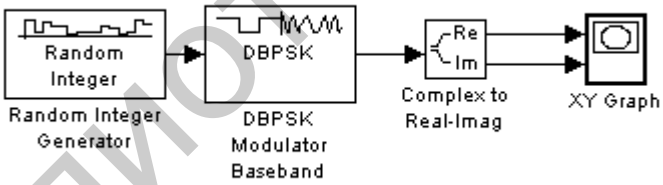


Рис. 38. Структурная схема модулирования сигнала с DQPSK в SIMULIK

Приведем более подробные сведения сначала для моделирования бинарной фазовой манипуляции (Binary Phase Shift Keying, BPSK), а затем в соответствии с появившимся уже достаточно заметным опытом работы с пакетом MATLAB/SIMULINK и с другими схемами фазовой манипуляции, которым посвящена эта лабораторная работа.

1. Войдем в программу SIMULINK и выберем последовательно следующие опции **File –> New –> Model** в **Simulink Library Browser**, чтобы сформировать новую модель.

2. Выберем опции **Communications Blockset** -> **Comm Sources** -> **Random Data Sources**, а затем выберем блок **Random Integer Generator** в модельном окне программы. Осуществив двойной щелчок мышью на этом блока, установим следующие его параметры:

- M−ary number to 2
- Initial seed to 37
- Sample time to 0.1
= Output Data Type to double

3. Выберем опции **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** > **PM** и из открывшегося окна установим блок **BPSK Modulator Baseband** в модельном окне программы.

4. Для иллюстрации выполненных действий приведем рис. 39, на котором представлена папка **Simulink Library Browser**

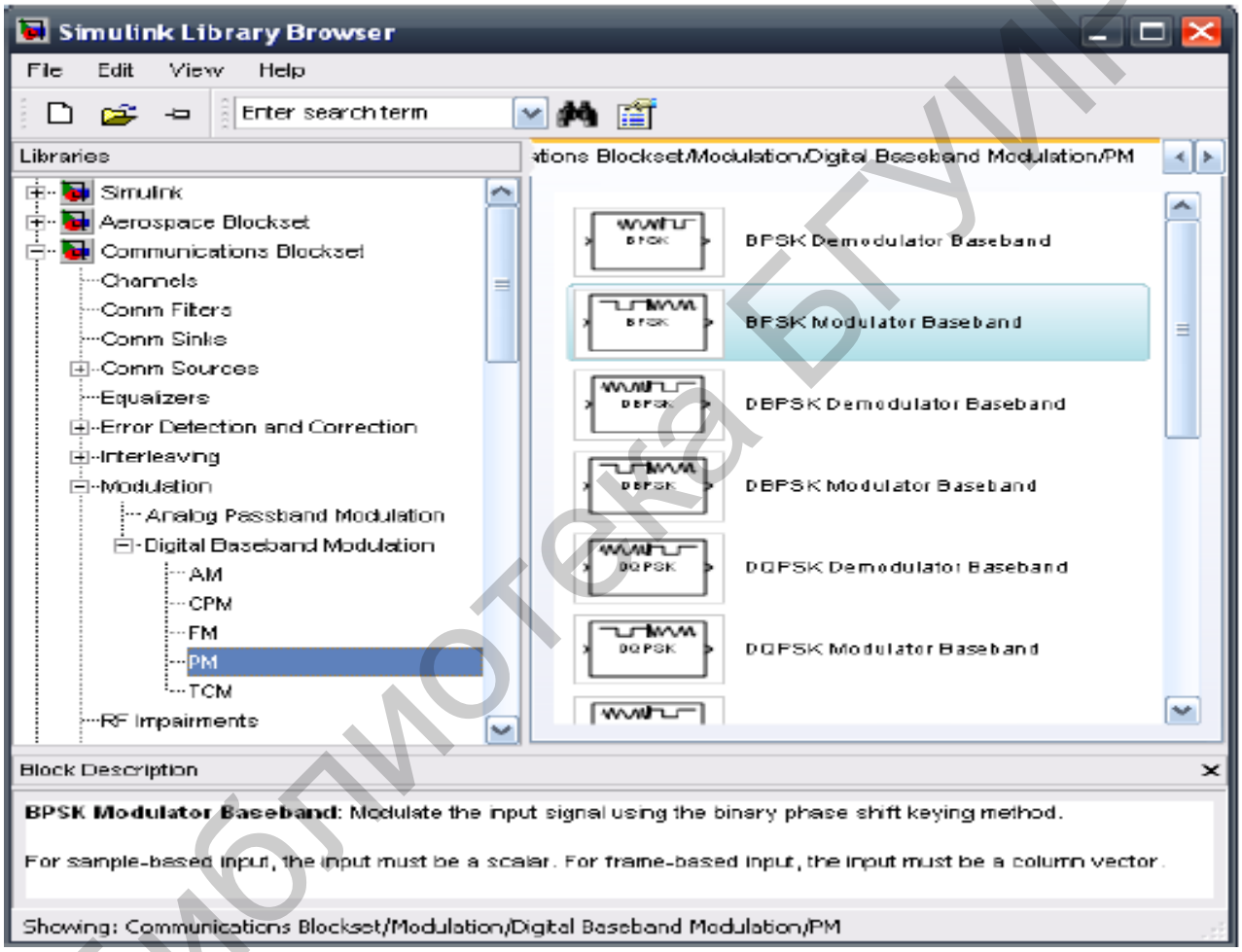


Рис. 39. Общий вид папки **Simulink Library Browser**

5. Выполним опции **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** > **PM** и установим блок **BPSK Demodulator Baseband** в модельном окне программы.

6. Теперь можно воспользоваться почти всеми теми же модулями, которые использовались до сих пор при выполнении данной лабораторной работы.

7. После запуска программ на выполнение нужно сохранить все диаграммы и значения в модуле **Display**.

Исследование процессов при квадратурной фазовой манипуляции также начнем со входа в программу **Simulink**, затем выполнить нужно следующее.

1. Выберем опции **File** -> **New** -> **Model**, чтобы сформировать новую модель для исследования. Папка **Simulink Library Browser** приведена на рис. 40.

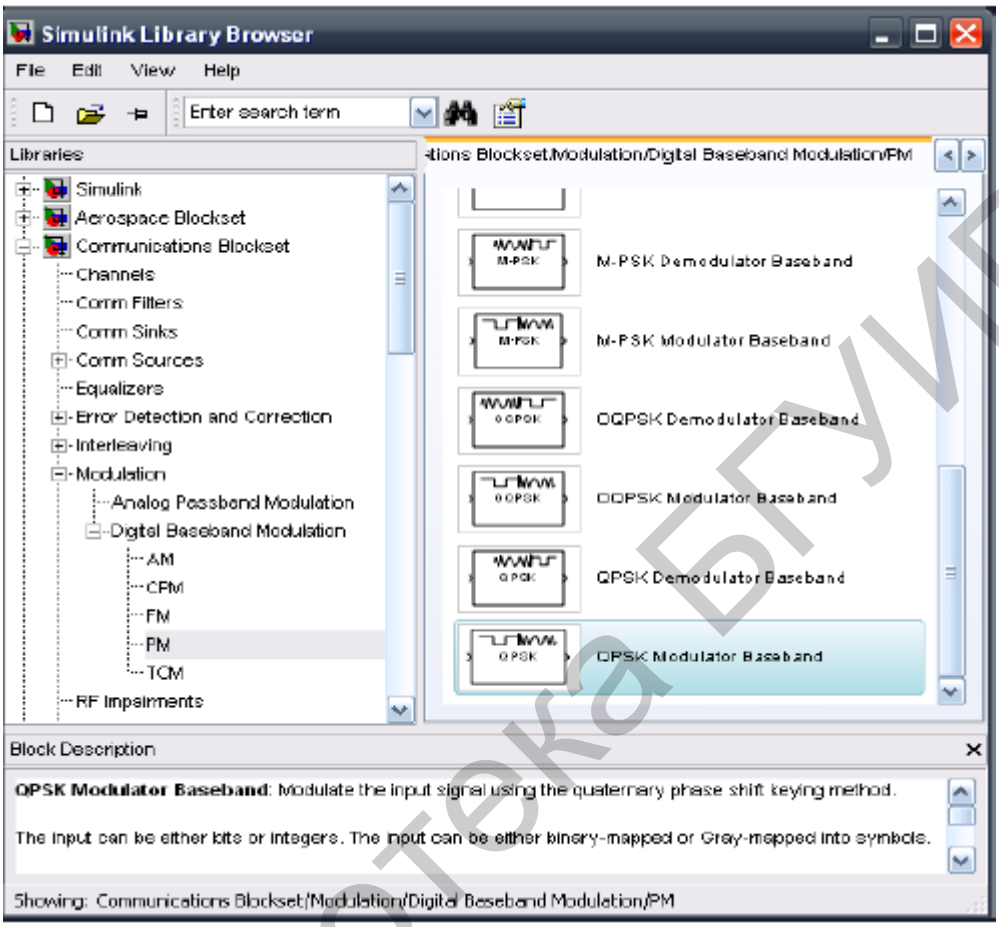


Рис. 40. Общий вид папки **Simulink Library Browser**, из которой следует выбрать модуль **QPSK Modulator Baseband**

2. Перейдем к **Communications Blockset** -> **Comm Sources** -> **Random Data Sources**, а затем выберем блок **Random Integer Generator** в модельном окне программы. Осуществив двойной щелчок мышью на этом блока, установим следующие его параметры:

- M−ary number to 4
- Initial seed to 37
- Sample time to 0.1
- Output Data Type to double.

3. Перейдем к **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** > **PM**, перетащим модуль **QPSK Modulator Baseband** в модельное окно программы.

4. Снова перейдем к **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** > **PM**, перетащим модуль **QPSK Demodulator Baseband** в модельное окно программы.

5. После этого нужно выполнить уже знакомые операции и сохрнить все построенные диаграммы и полученные значения измеряемых величин и осмыслить все экспериментальные данные и выводы записать в отчет по лабораторной работе.

Сделаем несколько пояснений, которые могут оказаться полезными при выполнении лабораторной работы в части изучения фазовой М-ичной манипуляции.

1. Войдем в программу Simulink. Затем сформируем новую модель для исследования, набрав опции **File** -> **New** -> **Model.**

2. Перейдем к **Communications Blockset** -> **Comm Sources** -> **Random Data Sources**, установим модуль **Random Integer Generator** в модельном окне программы. Двойным щелчком мыши установим следующие параметры этого модуля:

-M−ary number to 2

- Initial seed to 37

- Sample time to 0.1

- Output Data Type to double.

3. Перейдем к **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** -> **PM**, установим модуль **M-PSK Modulator Baseband** в модельном окне программы, как это показано на рис. 41. Двойным щелчком мыши установим следующие параметры этого модуля, как это видно из папки, приведенной на рис. 42:

= M−ary number to 2

- Input type to Integer

- Constellation ordering to Binary

- Phase offset (rad) to pi/8

- Output Data Type to double.

4. Теперь осталось выполнить почти те же операции, которые уже неоднократно выполнялись на протяжении выполнения этой лабораторной работы. После выполнения этой части работы нужно сохранить все диаграммы и значения измеренных величин, осмыслить их и сделать выводы, которые следует занести в отчет о выполненной работе. При этом нужно изменить М-ичные числа до 4, 8, 16, 32 и 64, а также параметры модулей **Random Integer Generator**, **M-PSK Modulator Baseband**, **M-PSK Demodulator Baseband**.

.

.

.

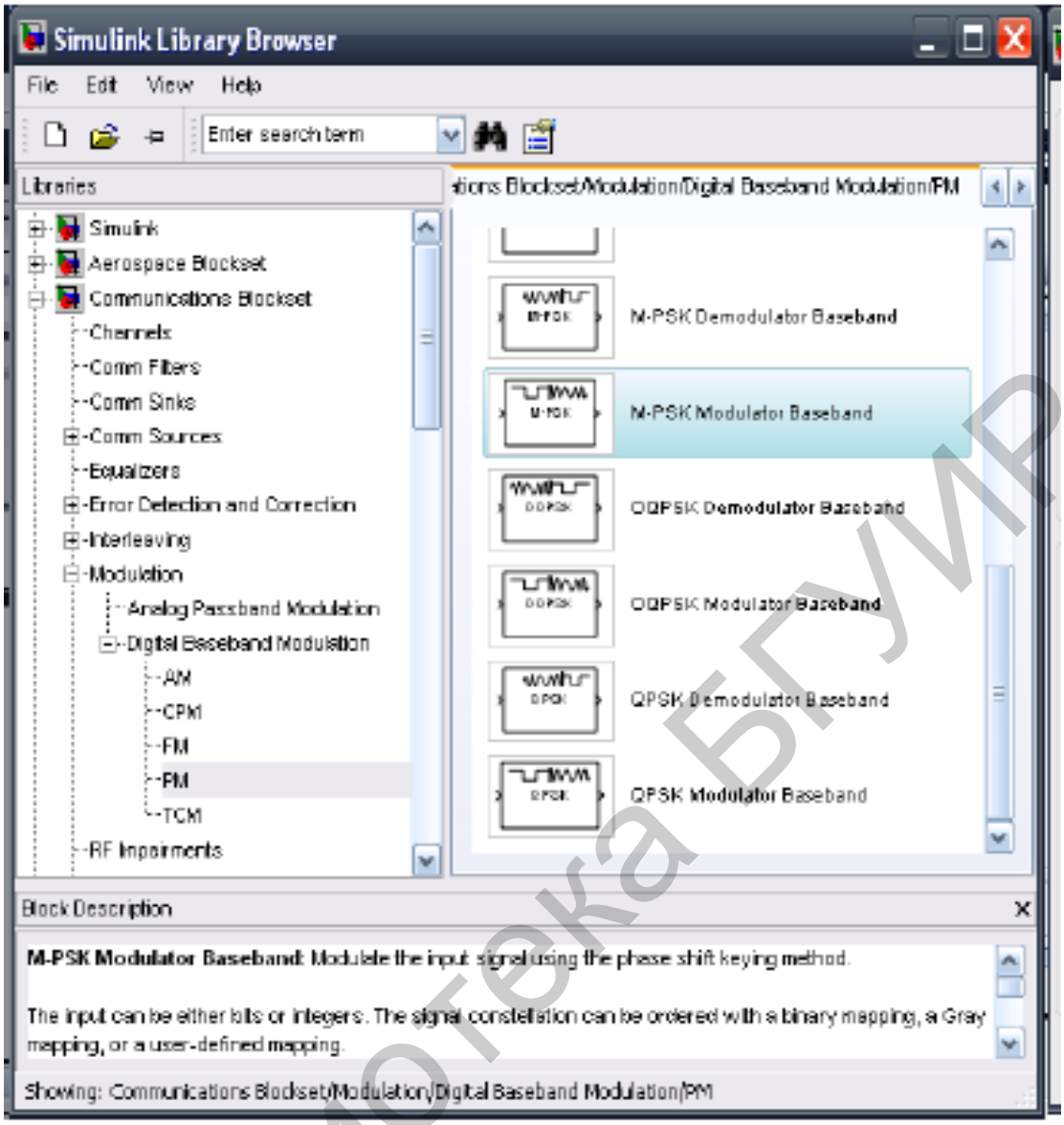


Рис. 41. Общий вид папки **Simulink Library Browser**, из которой будет выбран модуль **M-PSK Modulator Baseband**

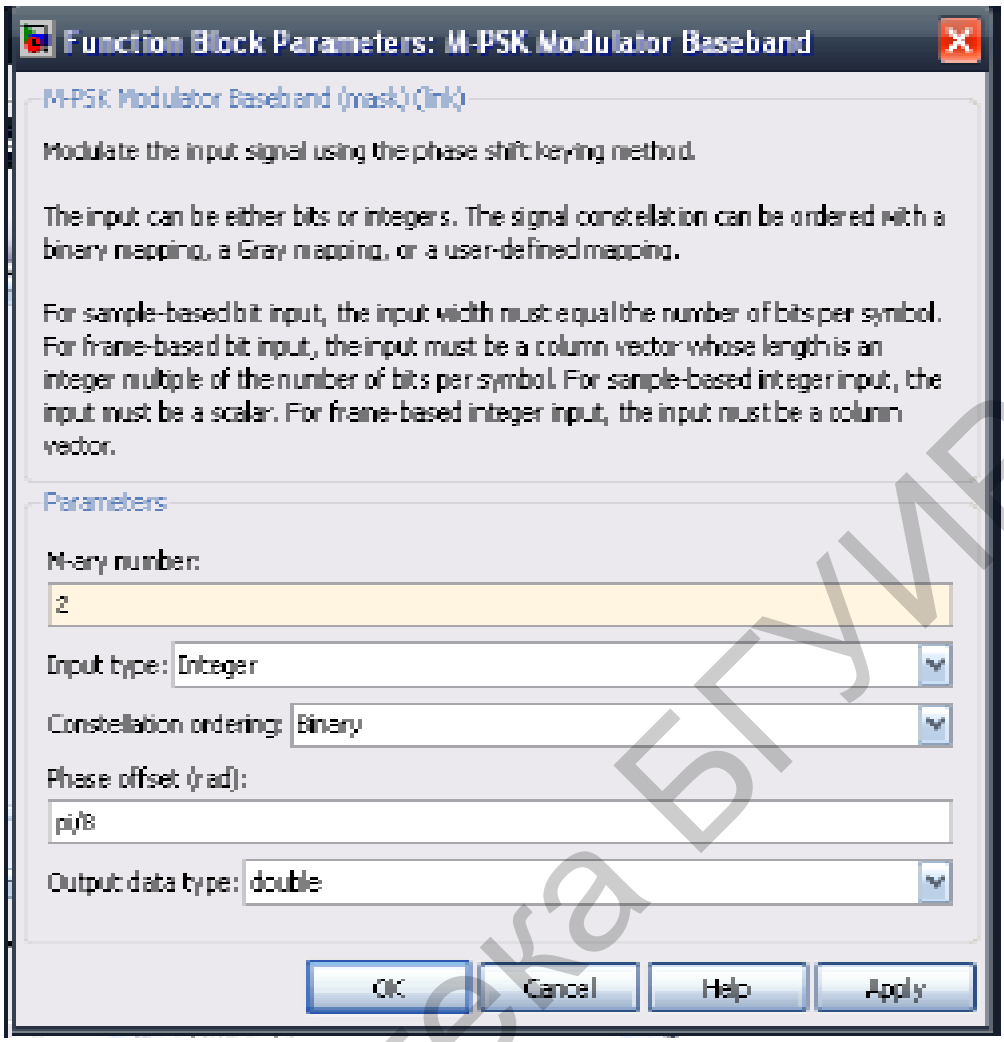


Рис. 42. Общий вид папки **Function Block Parameters: M-PSK Modulator Baseband**

И, наконец, несколько слов о моделировании сигнала с QAM и SIMULINK. Структурная схема имеет вид, показанный на рис. 43.

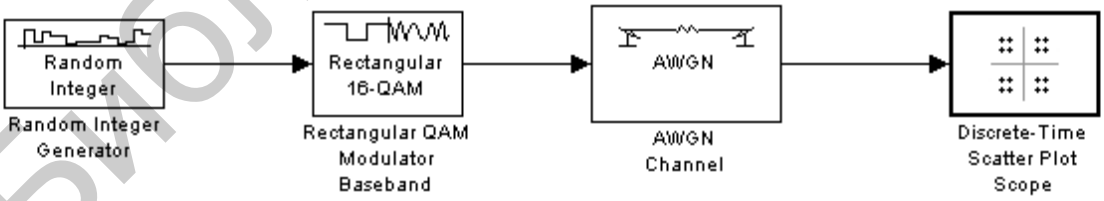


Рис. 43. Структурная схема модулирования сигнала с QAM в SIMULINK

1. Войдем в программу Simulink. Затем сформируем новую модель для исследования, набрав опции **File** -> **New** -> **Model.**

2. Перейдем к **Communications Blockset** -> **Comm Sources** -> **Random Data Sources**, установим модуль **Random Integer Generator** в модельном окне программы, как это показано на рис. 44. Двойным щелчком мыши установим следующие параметры этого модуля, как это видно из папки, приведенной на рис. 45:

\ - M−ary number to 2
- Initial seed to 37
- Sample time to 0.1
- Output Data Type to double.

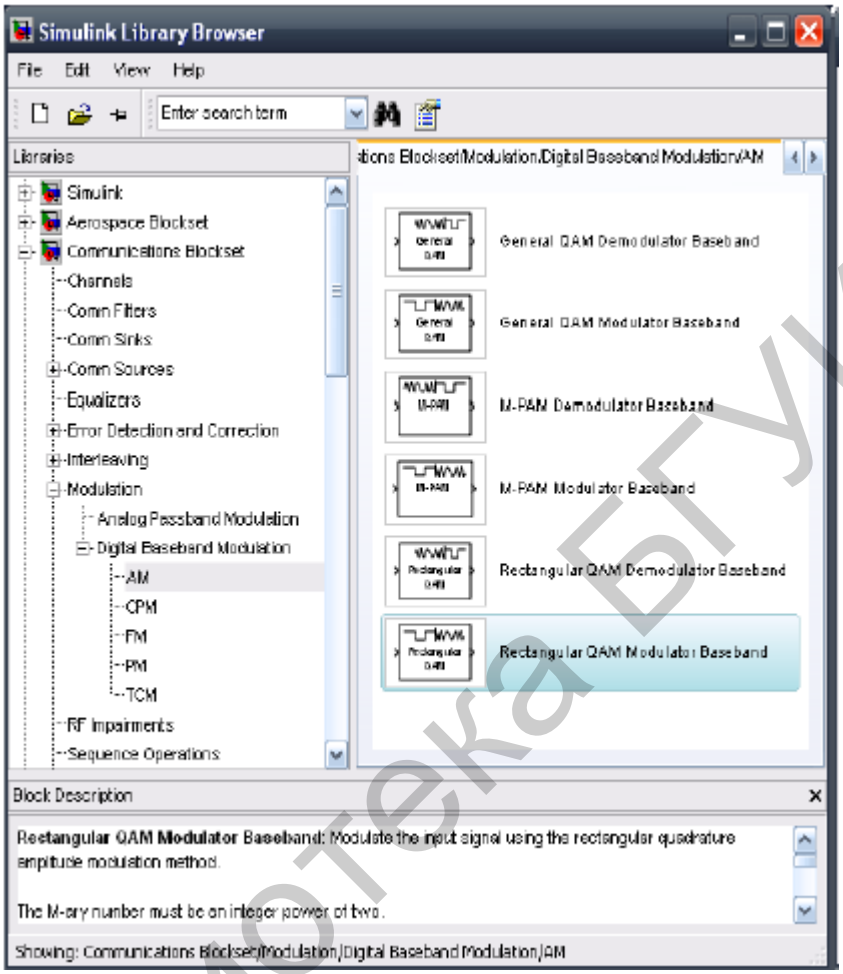


Рис. 44. Общий вид папки **Simulink Library Browser**, из которой будет выбран модуль **Rectangular QAM Modulator Baseband**

3. Перейдем к **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** > **AM**, установим модуль **Rectangular QAM Modulator Baseband** в модельном окне программы. Двойным щелчком мыши установим следующие параметры этого модуля, как это видно из папки, приведенной на рис. 45:

= M−ary number to 2
- Input type to Integer
- Constellation ordering to Binary
- Normalization method to Peak Power
- Peak power (watts): to 1
- Phase offset (rad) to 0
- Output Data Type to double

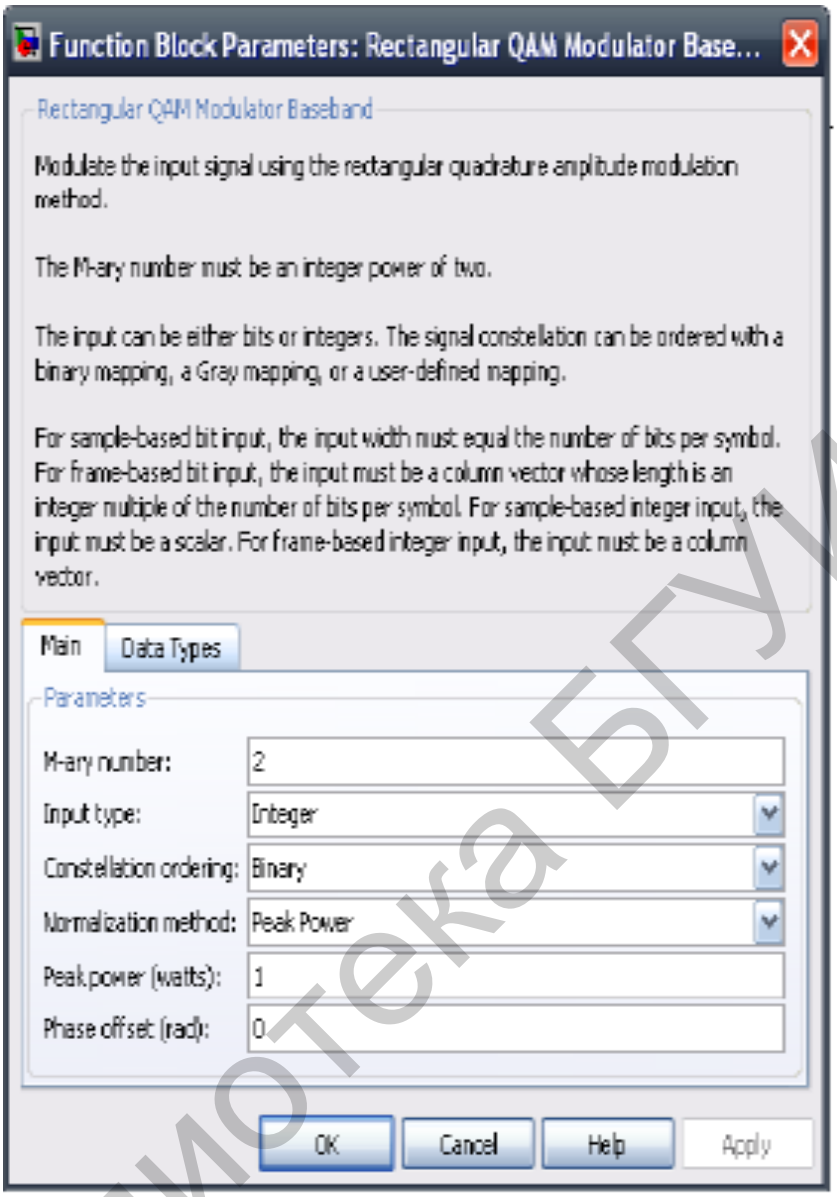


Рис. 45. Общий вид папки **Function Block Parameters: Rectangular QAM Modulator Baseband**

4. Перейдем снова к **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** > **AM**, перетащим модуль **Rectangular QAM Demodulator Baseband** в модельное окно программы. Двойным щелчком мыши установим следующие параметры этого модуля:

- M−ary number to 2
- Output type to Integer
- Constellation ordering to Binary
- Normalization method to Peak Power
- Peak power (watts): to 1
- Phase offset (rad) to 0.

5. Теперь нужно выполнить почти все те же операции, которые неоднократно выполнялись в процессе выполнения этой лабораторной работы. После выполнения этой программы нужно сохранить все диаграммы и значения измеренных величин, осмыслить их и сделать свои выводы, которые следует занести в отчет о выполненной работе.

6. При выполнении лабораторной работы следует изменить М-ичные числа до 4, 8, 16, 32 и 64, изменив также параметры модулей **Random Integer Generator, Rectangular QAM Modulator Baseband** и **Rectangular QAM Demodulator Baseband**.

7. Проведите исследование работы схемы, приведенной на рис.46

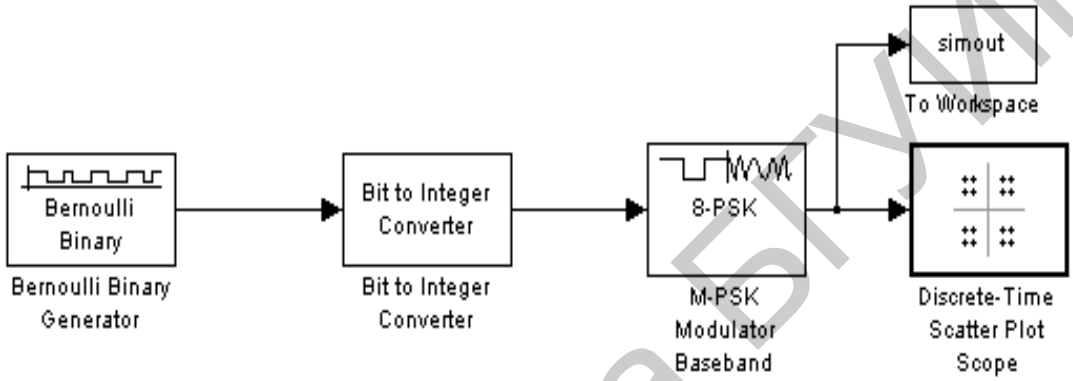


Рис. 46. Структурная схема модулирования сигнала с 8-PSK в SIMULINK

Выполните моделирование различных сигналов сначала применительно к каналу связи с отсутствием в нем шума, а затем введите различные уровни аддитивного белого шума и проанализируйте прохождение сигналов в этих условиях. Сформулируйте рассуждения по этим заданиям и сделайте свои выводы.

Выполните моделирование в Simulink схем, представленных на рис. 31… 52, напишите свои подробные комментарии к полученным результатам и сделайте общие выводы по всей лабораторной работе**.**

4. Контрольные вопросы

1. Для чего используется модуляция?
2. Что такое манипуляция и какие ее виды вам известны?
3. Что такое квадратурная манипуляция?
4. Назовите частные виды квадратурной манипуляции.
5. Что такое дифференциальная относительная фазовая манипуляция?
6. Назовите все известные вам созвездия сигналов.
7. Что такое величина BER, использующаяся в практике цифровой радиосвязи?
8. В чем заключается идея дифференциального кодирования?
9. Какими преимуществами обладает квадратурная амплитудная модуляция?

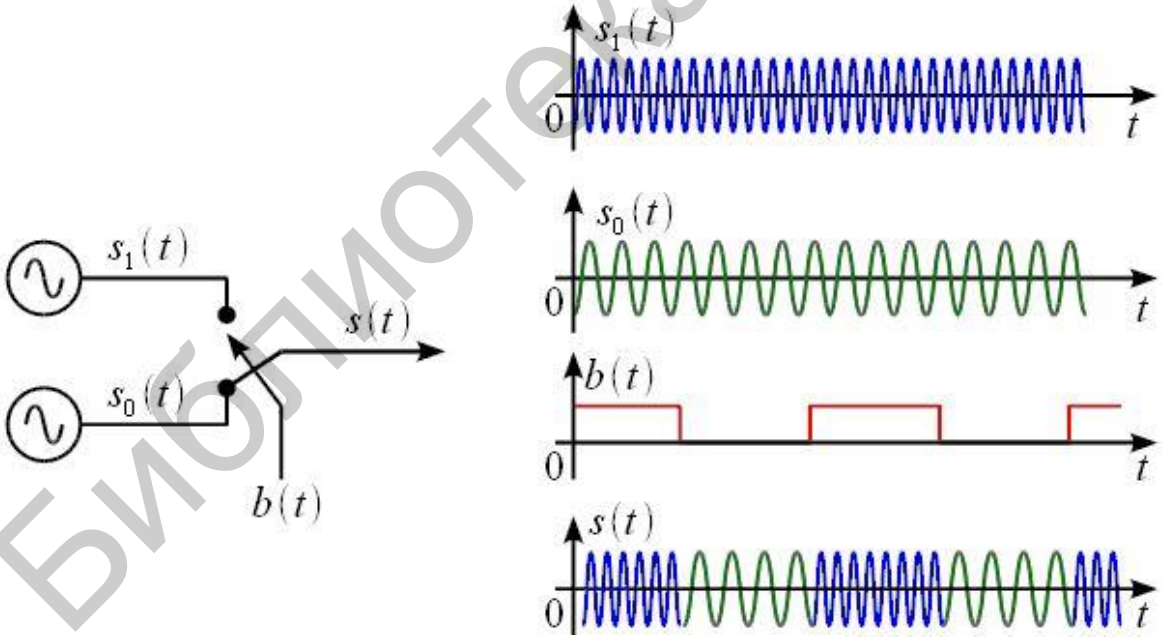
Лабораторная работа №5
ИССЛЕДОВАНИЕ ЧАСТОТНО-МАНИПУЛИРОВАННЫХ СИГНАЛОВ В СИСТЕМЕ MATLAB-SIMULINK

1. Краткие теоретические сведения

Частотно-манипулированные (Frequency Shift Keying, FSK) сигналы одни из самых распространенных в современной цифровой связи. Это обусловлено прежде всего простотой их генерирования и приема ввиду нечувствительности к начальной фазе. В данной работе проводится исследование принципа формирования бинарных сигналов FSK с различными параметрами. В русскоязычной литературе также встречается аббревиатура «ЧМн» для обозначения частотно-манипулированных сигналов.

1.1. Бинарная частотная манипуляция

Для начала рассмотрим двоичную FSK-модуляцию, когда исходный модулирующий сигнал $b(t)$ представляет собой бинарную последовательность нулей и единиц, следующую с битовой скоростью Br. Формирователь FSK-сигнала и принцип его функционирования можно условно представить, как это показано на рис. 1.



а б

Рис. 1. Формирователь FSK-сигнала (а) и временные диаграммы (б)

На рис. 1 показаны два генератора, формирующие колебания $s_0(t)$ и $s_1(t)$ на различных частотах. Также имеется электронный ключ, управляемый цифровым сигналом $b(t)$ таким образом, что при передаче логической 1 на выход подается сигнал $s_1(t)$, а при передаче логического 0 – сигнал $s_0(t)$. Таким образом, частота вы-

ходного сигнала изменяется в зависимости от битовой последовательности. Несмотря на простоту приведенной схемы, она на практике не применяется, поскольку требуется очень быстродействующий ключ с минимальным переходным процессом, а также при произвольной начальной фазе генераторов возможны скачки по фазе при смене символа, что в свою очередь приводит к расширению спектра. На практике получила распространение FSK-модуляция с непрерывной фазой CPFSK. Рассмотрим данный вид модуляции более подробно. FSK-сигналы являются частным случаем сигналов с частотной модуляцией при модулирующем сигнале в виде двоичной битовой последовательности $b(t)$. Таким образом, для модуляции FSK можно использовать схему частотного модулятора на базе универсального квадратурного модулятора, как это показано на рис. 2.

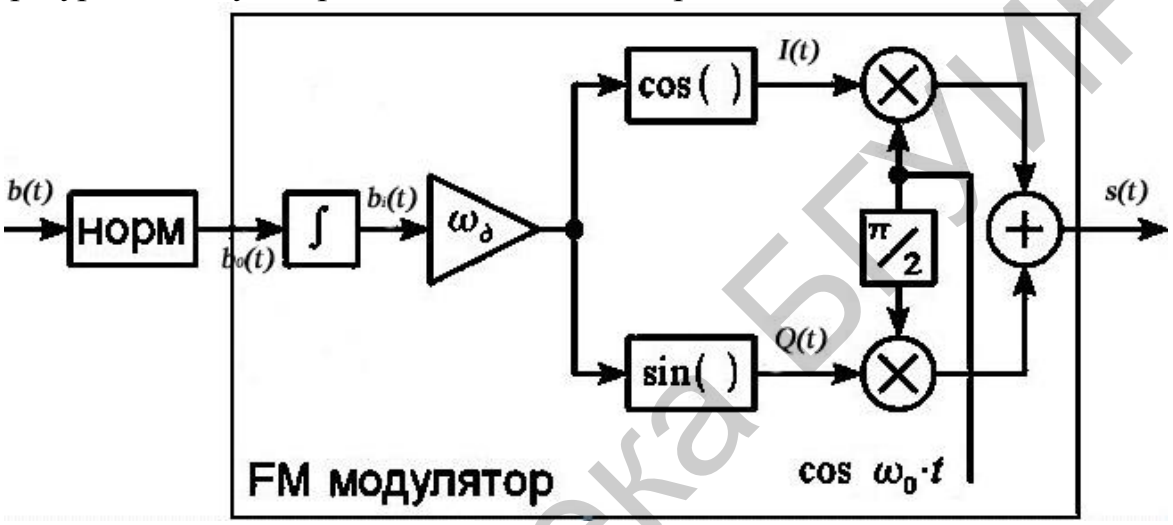


Рис. 2. Структурная схема формирования FSK-сигнала с помощью частотного модулятора

Поясняющие графики работы приведенной на рис. 2 структурной схемы показаны на рис. 3.

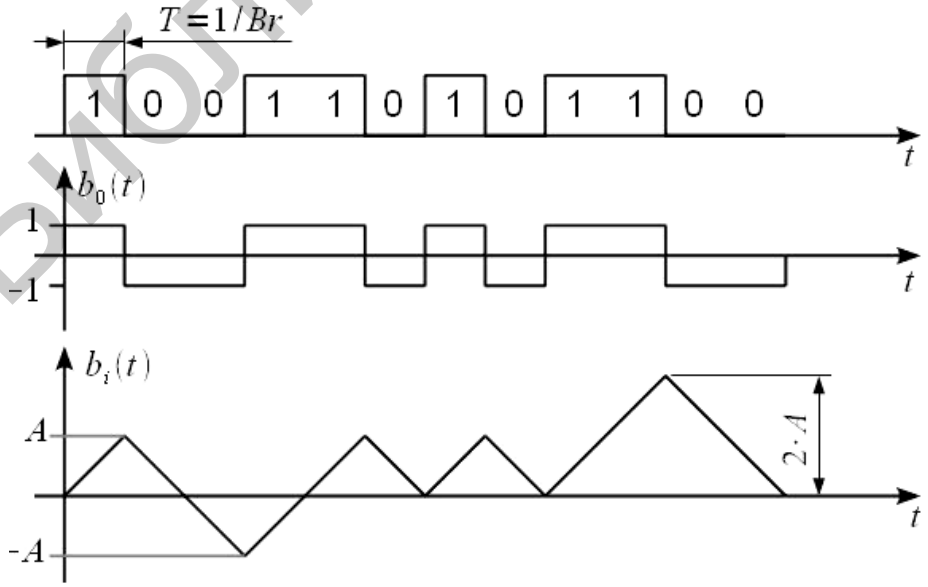


Рис. 3. Поясняющие графики работы FSK-модулятора

На верхнем графике показана исходная битовая последовательность $b(t)$, следующая со скоростью Br бод, т. е. длительность одного бита последовательности $T=1/Br \cdot$ с. Блок нормировки формирует сигнал $b_0(t)$ с уровнем ±1 и с нулевым средним, как это показано на среднем графике рис. 3, при этом форма сигнала сохраняется. Далее $b_0(t)$ используется как модулирующий сигнал на входе частотного модулятора. Первым блоком частотного модулятора стоит интегратор, который интегрирует сигнал $b_0(t)$ в результате получается сигнал $b_i(t)$ в виде пилообразного напряжения, как это показано на нижнем графике рис. 3. Необходимо отметить, что при интегрировании импульс единичной амплитуды на выходе интегратора будет иметь амплитуду$A=T=1/Br$. После этого сигнал на выходе интегратора $b_i(t)$ усиливается в $\omega_д$ раз, где $\omega_д$ – частота девиации FM-сигнала. При рассмотрении FM-сигналов говорилось, что частота девиации задает полосу сигнала на выходе модулятора. При цифровой модуляции частота девиации задает разнос частот манипуляции. Представим $\omega_д$ в виде произведения:

$$\omega_д = \underbrace{2*\pi*Br/2}_{\Omega=2*\pi*F_b} * m = \pi*Br*m \text{ рад/с}, \qquad (1)$$

где m носит название индекса FSK-модуляции и определяет, во сколько раз разнос частот манипуляции превышает битовую скорость; $\Omega=2*\pi*F_b$ − циклическая частота модулирующего сигнала; $F_b = Br/2$ − частота повторения бита при чередовании нулей и единиц в цифровом сигнале (в 2 раза ниже скорости передачи информации Br). После усиления и задания девиации частоты производится формирование квадратурных компонент $I(t)$ и $Q(t)$ и модуляция при помощи универсального квадратурного модулятора.

Замечание. Смысл сигнала на выходе интегратора не что иное, как мгновенная фаза FSK-сигнала. Поскольку на выходе интегратора фаза не имеет разрывов, то формируемый таким образом FSK-сигнал называется FSK-сигнал с непрерывной фазой или CPFSK. Также в некоторой литературе такой способ модуляции носит название модуляция с памятью, так как интегратор «помнит» значения, полученные ранее, в то время как ключ на рис. 1 « не помнит » свое положение в предыдущие моменты времени (модулятор на рис. 1 носит название модулятор без памяти).

1.2. Спектр FSK-сигнала

Рассмотрим спектр FSK-сигнала. Спектр сигналов с угловой модуляцией в общем случае не выражается аналитически. Однако в случае с бинарной последовательностью можно получить оценку спектра FSK-сигналов, следуя следующим рассуждениям. Представим сигнал $b_0(t)$ на входе FM-модулятора в виде суммы двух сигналов:

$$b_0(t) = b_L(t) + b_H(t),$$

$$\text{где } b_H(t) = \begin{cases} -1, & b_0(t) > 0 \\ 0, & b_0(t) < 0 \end{cases}, \qquad (2)$$

$$b_L(t)=\begin{cases}-1, & b_0(t)<0\\ 0, & b_0(t)>0\end{cases} \qquad (3)$$

Такое представление графически показано на рис. 4.

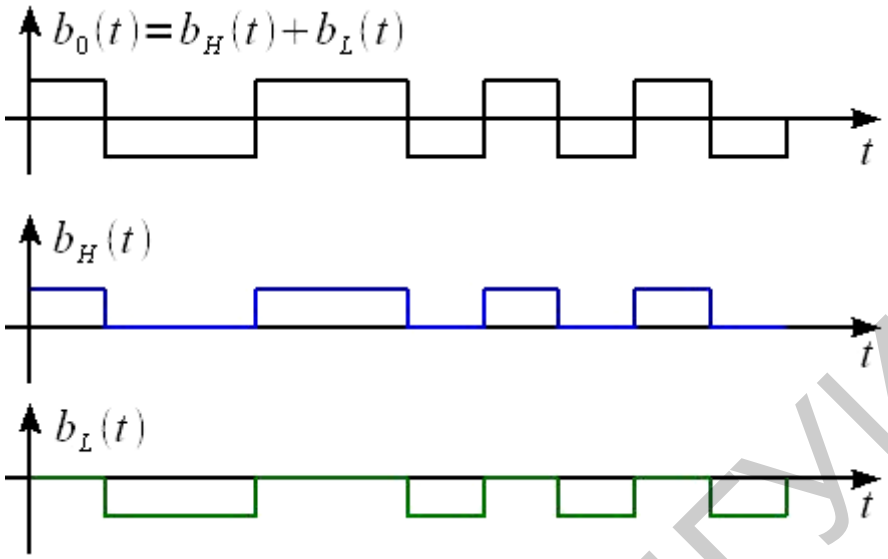


Рис. 4. Представление модулирующего сигнала

Тогда FSK-сигнал можно представить в виде суммы сигналов $s_L(t)$ и $s_H(t)$:

$$s(t) = s_L(t) + s_H(t),$$
$$s_L(t) = b_L(t)\cos((\omega_0 - \omega_\partial)*t), \qquad (4)$$
$$s_H(t) = b_H(t)\cos((\omega_0 - \omega_\partial)*t).$$

Графически это показано на рис. 5.

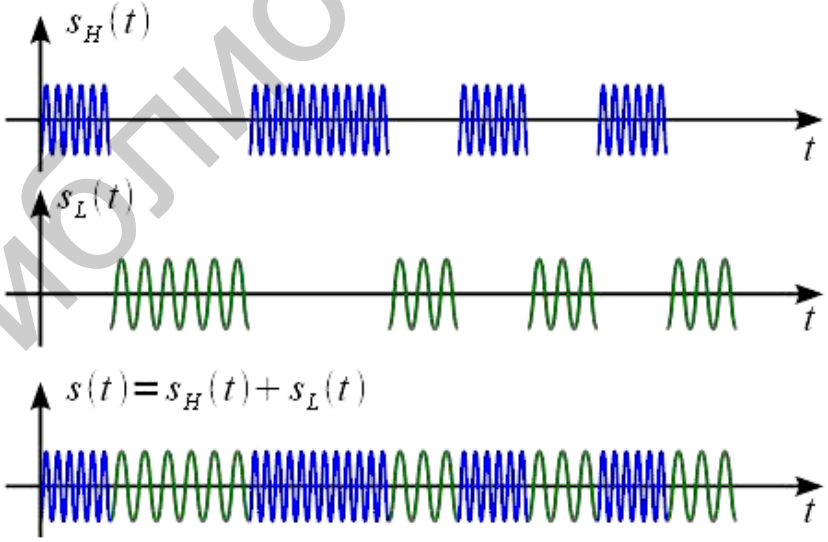


Рис. 5. Представление FSK-сигнала

Таким образом, спектр FSK-сигнала $S(\omega)$ есть сумма спектров $S_L(\omega) + S_H(\omega)$, сигналов $s_L(t)$ и $s_H(t)$. Но согласно (4) $s_L(t)$ и $s_H(t)$ – перенесенные на

соответствующие частоты сигналы $b_L(t)$ и $b_H(t)$, которые в свою очередь представляют собой последовательность импульсов длительностью $T=1/Br$. Поскольку битовая последовательность случайная, то спектральная плотности $B_H(\omega)$ и $B_L(\omega)$ сигналов $b_L(t)$ и $b_H(t)$ может быть представлена, как это показано на рис. 6.

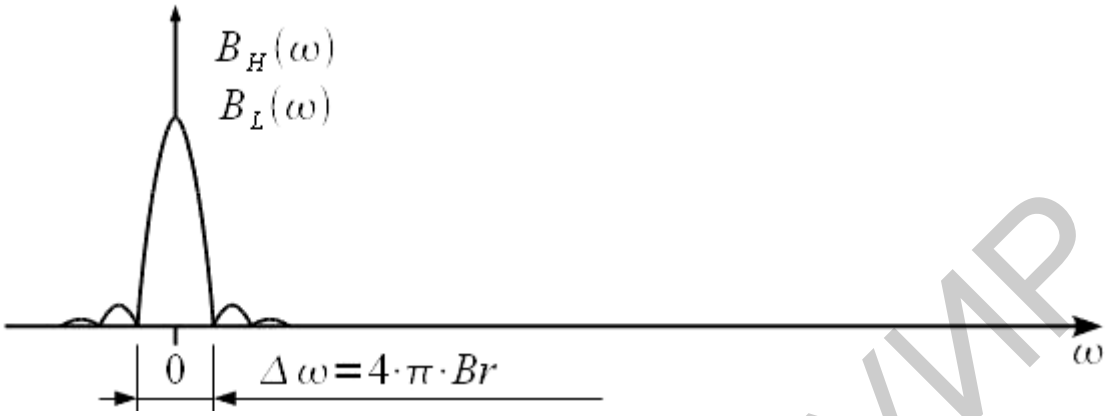


Рис. 6. Спектральная плотность случайного битового потока

Тогда спектры $S_L(\omega)$ и $S_H(\omega)$ сигналов $s_L(t)$ и $s_H(t)$, а также результирующий спектр FSK-сигнала представлены на рис. 7.

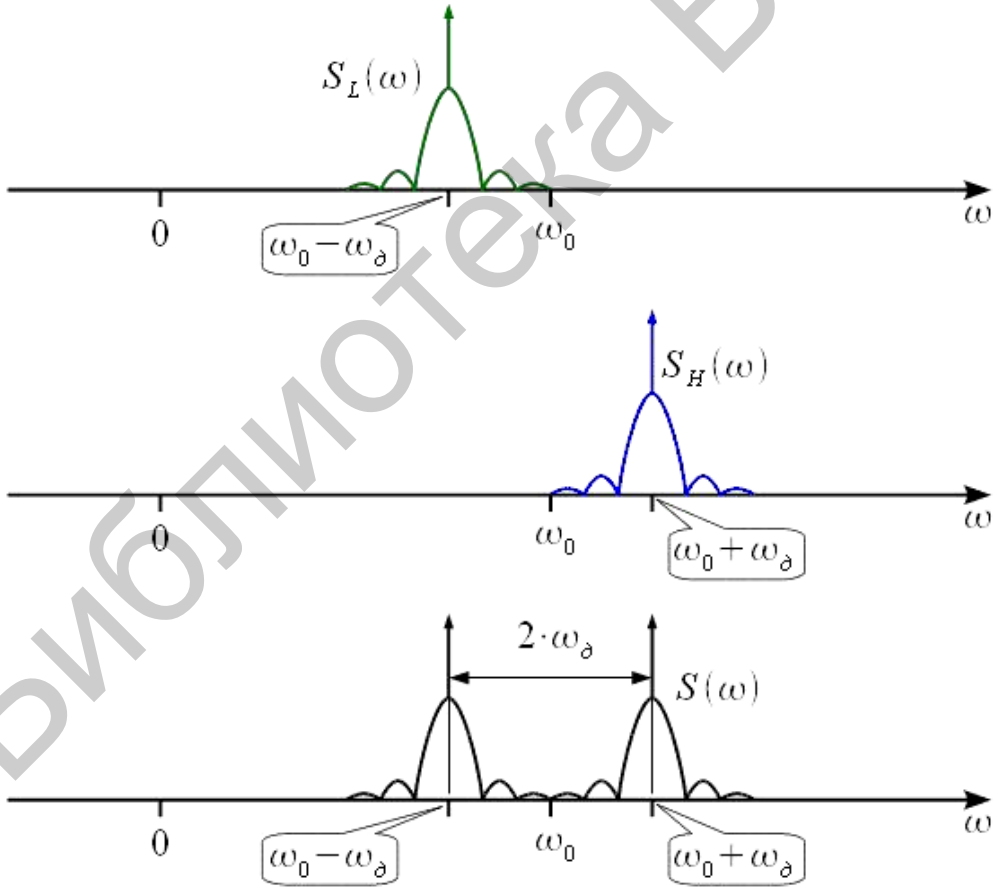


Рис. 7. Спектр FSK-сигнала

Таким образом, имеем спектр FSK-сигнала. Видно, что составляющие FSK-сигнала разнесены на частоту девиации, а согласно (1), частота девиации зависит от битовой скорости Br и индекса модуляции m. При фиксированной

битовой скорости составляющие спектра FSK-сигнала будут тем ближе друг к другу, чем меньше индекс FSK-модуляции. На рис. 8 показаны спектры FSK-сигнала при различном индексе модуляции.

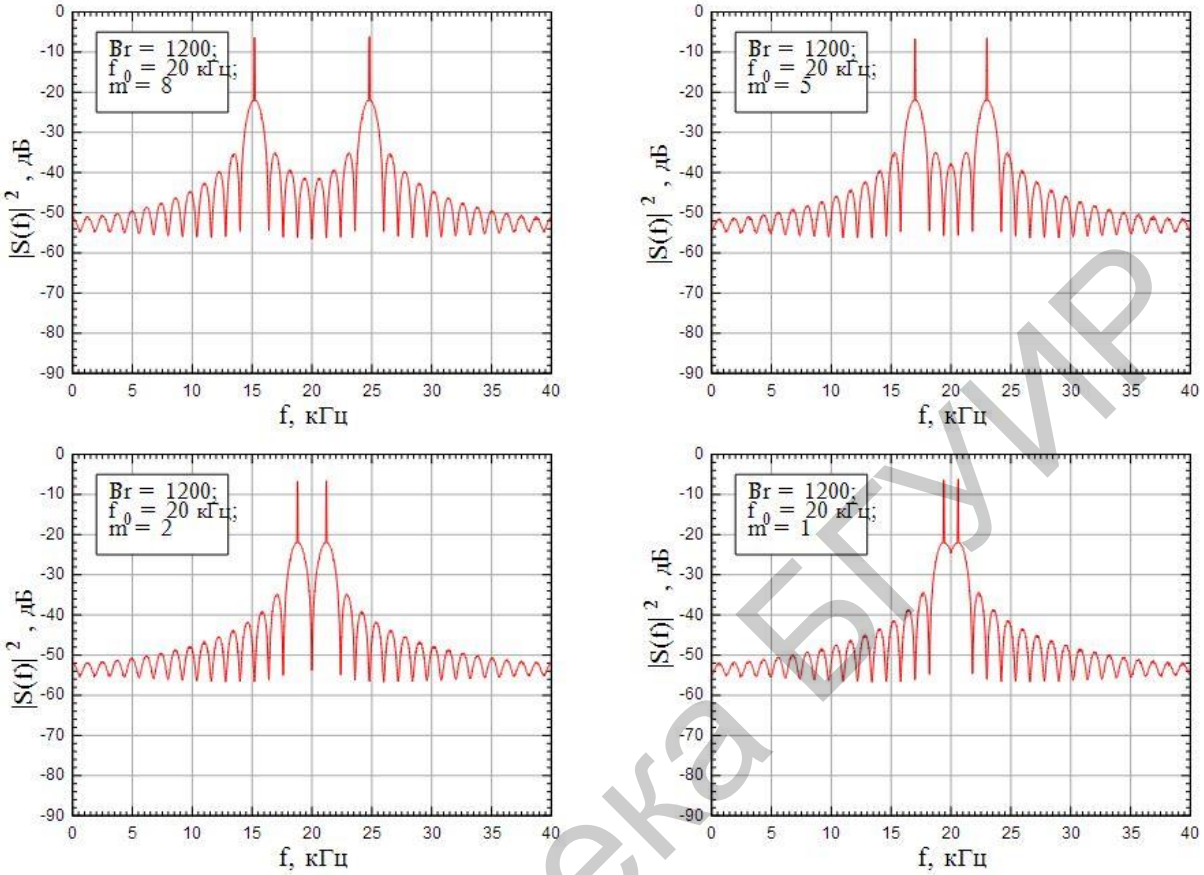


Рис. 8. Спектры FSK-сигнала при различном индексе модуляции

Из рис. 8 следует, что при уменьшении индекса FSK-модуляции составляющие FSK-сигнала сдвигаются и при $m = 2$ основные лепестки соприкасаются, а при $m = 1$ перекрываются наполовину. Таким образом, индекс модуляции задает положение составляющих FSK вне зависимости от несущей частоты и битовой скорости модулирующего сигнала.

На рис. 9 представлен спектр FSK и основные частотные соотношения.

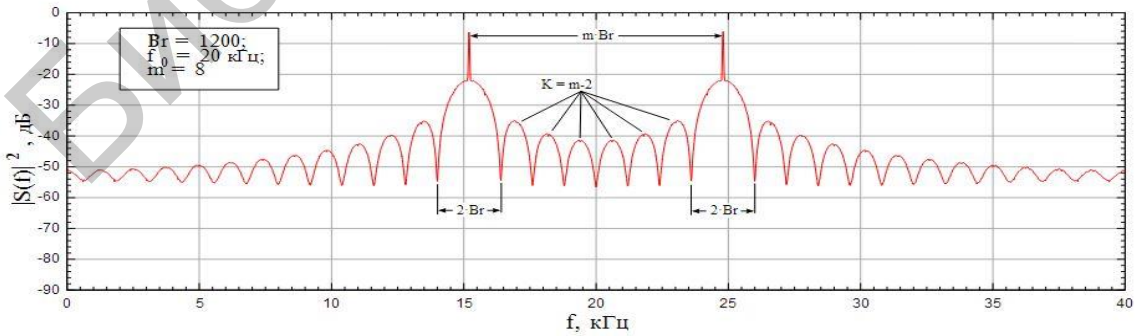


Рис. 9. Основные частотные соотношения в спектре FSK

Параметр K задает количество боковых лепестков между составляющими спектра.

2. Порядок выполнения лабораторной работы

2.1. Моделирование FSK-сигнала в пакете MATLAB

Рассмотрим пример моделирования FSK сигнала в пакете MATLAB. Ниже приведен код, который позволяет построить 4 графика: 1 и 2 – модулирующие несущее колебание (синусоида), 3 – импульсы бинарного сообщения, 4 – частотно-манипулированное колебание.

```
clc % Для очистки командной строки
close all % Закрываем все окна, кроме командной строки
clear all % Удаляем все переменные из памяти
fc1=input(' Введите значение частоты для 1-го синусоидального носителя:');
fc2=input(' Введите значение частоты для 2-го синусоидального носителя:');
fp=input(' Введите значение частоты для периодического бинарного импульса (Сообщение):');
amp=input(' Введите амплитуду (для обоих носителей и периодического бинарного импульса)');
amp=amp/2;
t=0:0.001:1; % Для установки интервала дискретизации
c1=amp.*sin(2*pi*fc1*t);% Для генерации волны первого синусоидального носителя
c2=amp.*sin(2*pi*fc2*t);% Для генерации волны второго синусоидального носителя
subplot(4,1,1); % Для построения несущей волны
plot(t,c1)
xlabel('Time')
ylabel('Amplitude')
title('Carrier 1 Wave')
subplot(4,1,2) % Для построения несущей волны
plot(t,c2)
xlabel('Time')
ylabel('Amplitude')
title('Carrier 2 Wave')
m=amp.*square(2*pi*fp*t)+amp;%Для генерации сигнала сообщения
subplot(4,1,3) % Для построения бинарного сигнала (сообщения)
plot(t,m)
xlabel('Time')
ylabel('Amplitude')
title('Binary Message Pulses')
for i=0:1000 % Здесь мы генерируем модулированную волну
   if m(i+1)==0
      mm(i+1)=c2(i+1);
      else
      mm(i+1)=c1(i+1);
   end
end
subplot(4,1,4) % Для построения модулированной волны
plot(t,mm)
xlabel('Time')
ylabel('Amplitude')
title('Modulated Wave')
```

Вводим значения для FSK:

1. Введите значение частоты для 1-го синусоидального носителя: 10
2. Введите значение частоты для 2-го синусоидального носителя: 30
3. Введите значение частоты для периодического бинарного импульса (Сообщение): 5
4. Введите амплитуду (для обоих носителей и периодического бинарного импульса): 4

Результат моделирования представлен на рис. 10.

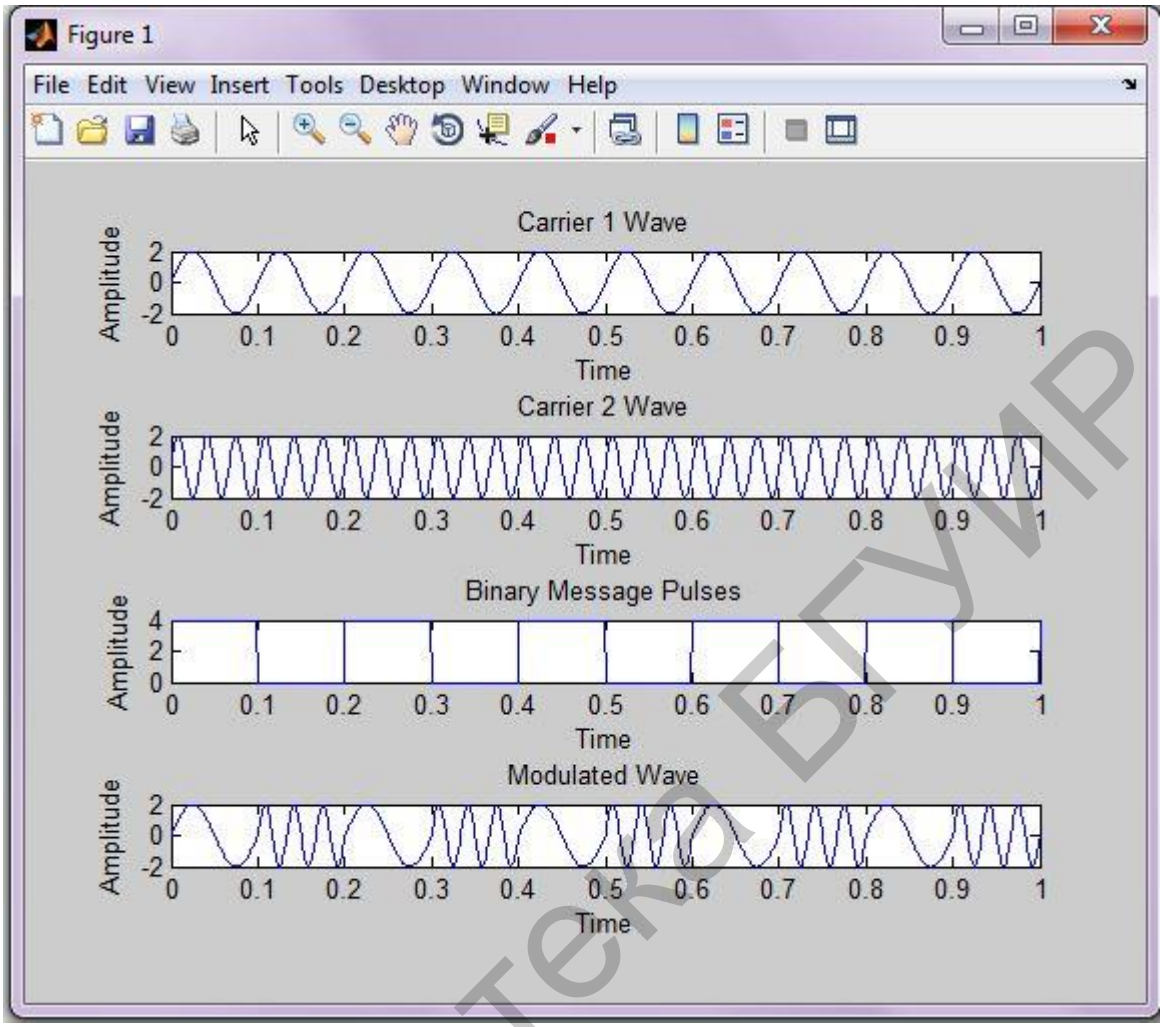


Рис. 10. Моделирование FSK в пакете MATLAB

Теперь самостоятельно измените несколько раз значения для FSK и зафиксируйте полученные результаты.

2.2. Анализ спектров и автокорреляционных функций частотно-манипулированных сигналов в среде MATLAB

Построим спектры и автокорреляционные функции (АКФ) частотно-манипулированных сигналов, заданные тремя манипулирующими функциями в виде m-последовательностей, полученных с помощью четырехразрядного регистра, охваченного цепью обратной связи.

Исходные данные – манипулирующие функции, представляющие собой m-последовательности, полученные при помощи 4-разрядного регистра сдвига, охваченного обратной связью. Длины последовательностей достигают 25 битов. Передаче высокого уровня (логической единицы) соответствует частота несущего колебания 2 кГц, передаче низкого уровня (логического нуля) – 1 кГц. Длительность одного элемента манипулирующей функции равна 1 мс,

т. е. на длительности одного элемента располагается либо один, либо два периода несущей частоты. Программа с пояснениями представлена ниже, а результат ее работы – на рис. 11.

```
>>bits=[1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 0 1 0]; %задаётся цифровое сообщение

>> N=length(bits); % вычисляется длина сообщения (в битах).

>> Fd=1e3; % скорость следования битов цифрового сообщения

>> FsFd=50;  %количество отсчетов дискретного времени, приходящихся на один %символ цифрового сообщения

>> Fs=Fd*FsFd; % производим вычисление частоты дискретизации

>> f=[1e3 2e3]; % задаем значения частот манипуляции 1000 Гц и 2000 Гц

>> t=(0:FsFd-1)/Fs; % рассчитываем дискретное время для одного символа %цифрового сообщения

>> pps=2*pi*f/Fs; % определяем сдвиг фазы на один отсчет

>> s1=repmat(pps,FsFd,1); %определяем фазовые сдвиги для логических нулей и единиц %цифрового сообщения. Результатом действия функции repmat является матрица, %содержащая два столбца, состоящих из FsFd элементов. В первом столбце будут %содержаться значения фазовых сдвигов для логических нулей, во втором – для %логических единиц

>> d_cpfsk=s1(:,bits+1); % производим выбор столбца, соответствующего текущему символу %сообщения. Итогом является формирование матрицы, содержащей 25 столбцов (25 бит %цифрового сообщения) по FsFd элементов

>> d_cpfsk=d_cpfsk(:); % все столбцы матрицы, сформированной предыдущей строкой, %выстраиваются подряд друг за другом. В результате получается матрица, состоящая из %одного столбца, содержащая все необходимые фазовые сдвиги

>> phi_cpfsk=cumsum(d_cpfsk); % выполняем интегрирование фазовых сдвигов для %получения фазовой функции

>> s_cpfsk=cos(phi_cpfsk); % формируем частотно-манипулированный сигнал

>> td=(0:N*FsFd-1)/FsFd; % организуем время для удобства отображения графика в %символах

>> plot(td,s_cpfsk) % выводим на экран график частотно-манипулированного сигнала во %временной области.
```

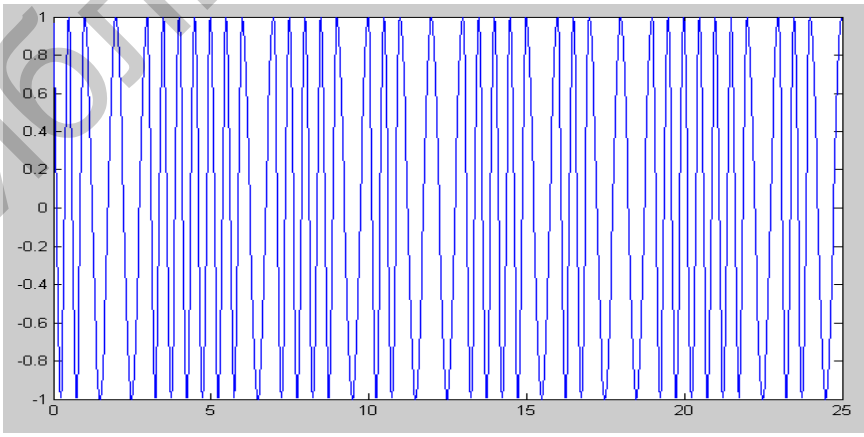


Рис. 11. Частотно-манипулированный сигнал во временной области для случая, когда манипулирующей функцией является

bits=[1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 0 1 0]

Такой способ формирования сигнала называется частотной манипуляцией с непрерывной фазовой функцией (continuous phase frequency shift keying, CPFSK). При этом формируется линейно меняющаяся полная фаза колебания, а передаваемые символы управляют скоростью ее изменения. При этом передаваемые символы переключают значение мгновенной частоты, которая интегрируется, давая непрерывную фазовую функцию. Косинус такой полной фазы тоже будет непрерывной функцией.

Следующим шагом является построение спектра частотно-манипулированного сигнала. Для этого в командной строке необходимо ввести следующую команду:

```
>> periodogram(s_cpfsk,[],[],Fs,'onesided') % данная функция (периодограмма) является функцией %непараметрического спектрального анализа. Она вычисляет и выводит на экран график %спектральной плотности мощности для одной (данной) реализации сигнала. При этом %спектр частотно-манипулированного сигнала, вычисленный вышеуказанным способом, %представлен на рис. 1.
```

Далее необходимо произвести построение АКФ частотно-манипулированного сигнала. Для этого в командной строке Matlab необходимо ввести команды:

```
>> K=length(td); % определяем число элементов дискретного времени, для которого будет %рассчитываться АКФ
>> [tmp,R]=corrmtx(s_cpfsk,K-1); %производится расчет корреляционной матрицы частотно-
%манипулированного сигнала
>> plot(td,R(1,:)) % выводим на экран первую строку этой матрицы, причем время для %графика также отображается в символах. В результате получаем АКФ частотно-%манипулированного сигнала, вычисленную таким способом.
```

Заключительным на данном этапе шагом является выполнение аналогичных действий при построении спектров и АКФ частотно-манипулированного сигналов для двух оставшихся манипулирующих функций. На рис. 12…16 приведены результаты выполнения этой работы для ряда манипулирующих функций, позволяющих провести анализ и обсуждение спектров и АКФ исследованных сигналов.

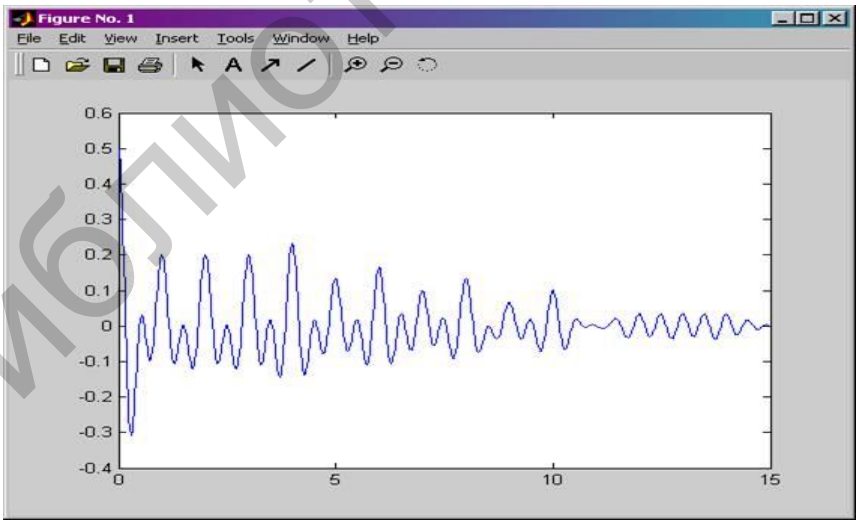


Рис. 12. АКФ частотно-манипулированного сигнала для случая, когда манипулирующей функцией является bits=[1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 0 1 0]

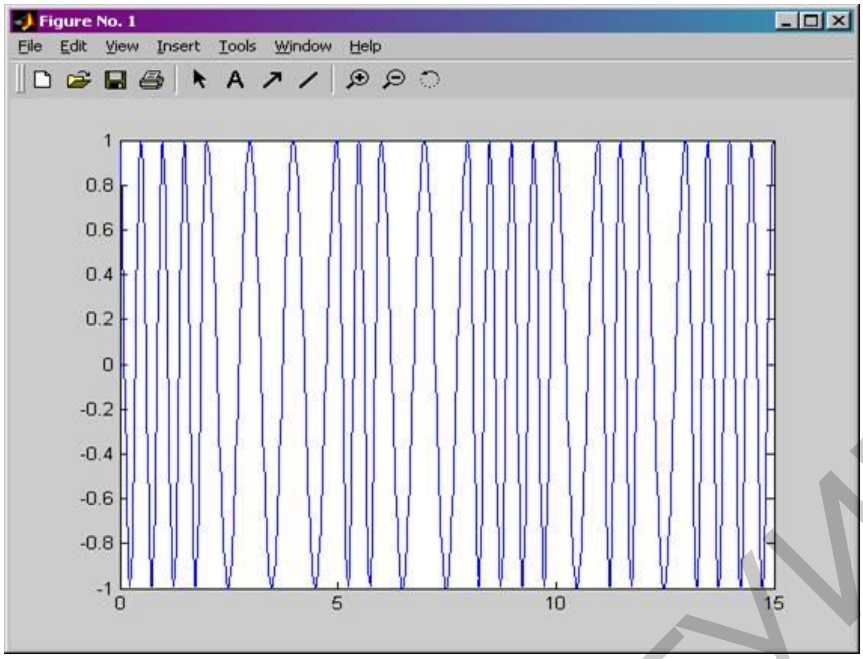


Рис. 13. Частотно-манипулированный сигнал во временной области для случая, когда манипулирующей функцией является [1 0 1 1 0 1 1 0 0 1 1 1 0 0 1 0 1]

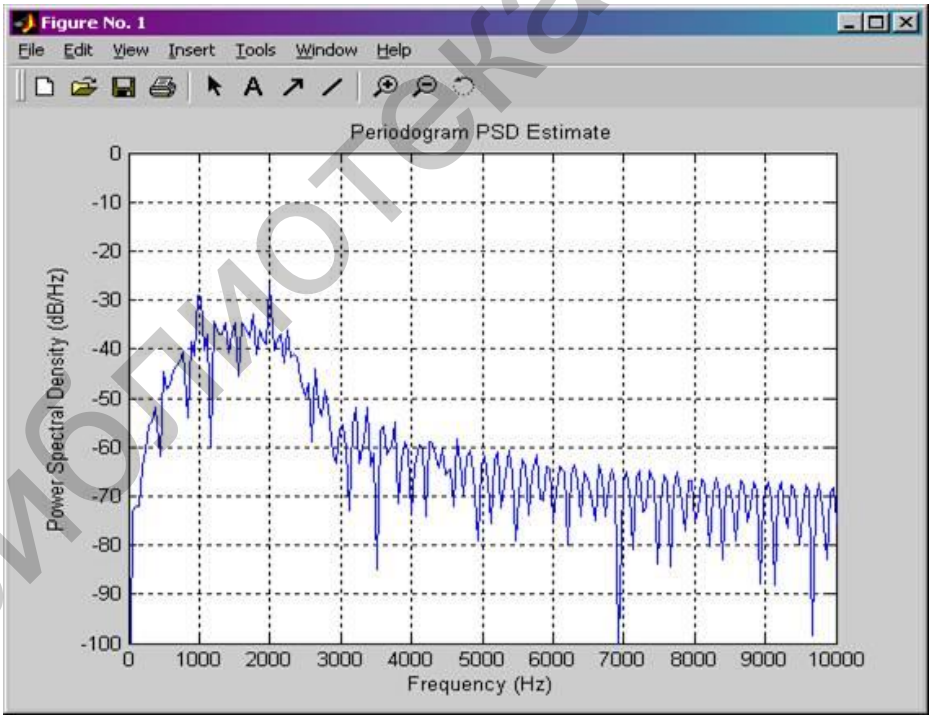


Рис. 14. Спектр мощности частотно-манипулированного сигнала для случая, когда манипулирующей функцией является [1 0 1 1 0 1 1 0 0 1 1 1 0 0 1 0 1]

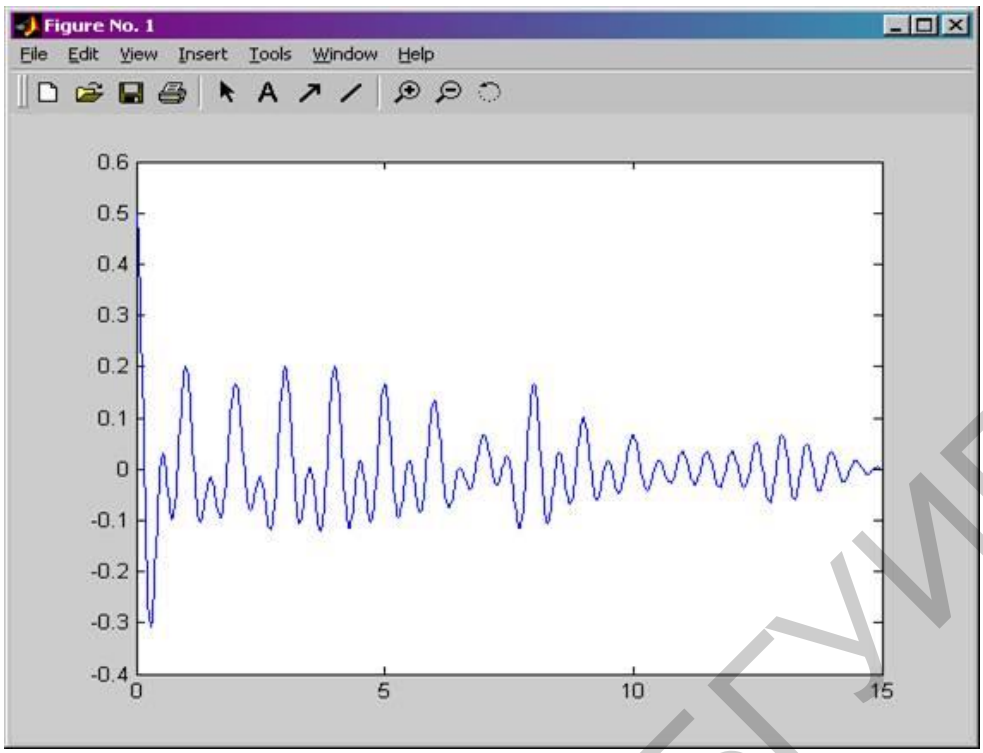


Рис. 15. АКФ частотно-манипулированного сигнала для случая, когда манипулирующей функцией является [1 0 1 1 0 11 0 0 1 1 1 0 0 1 0 1]

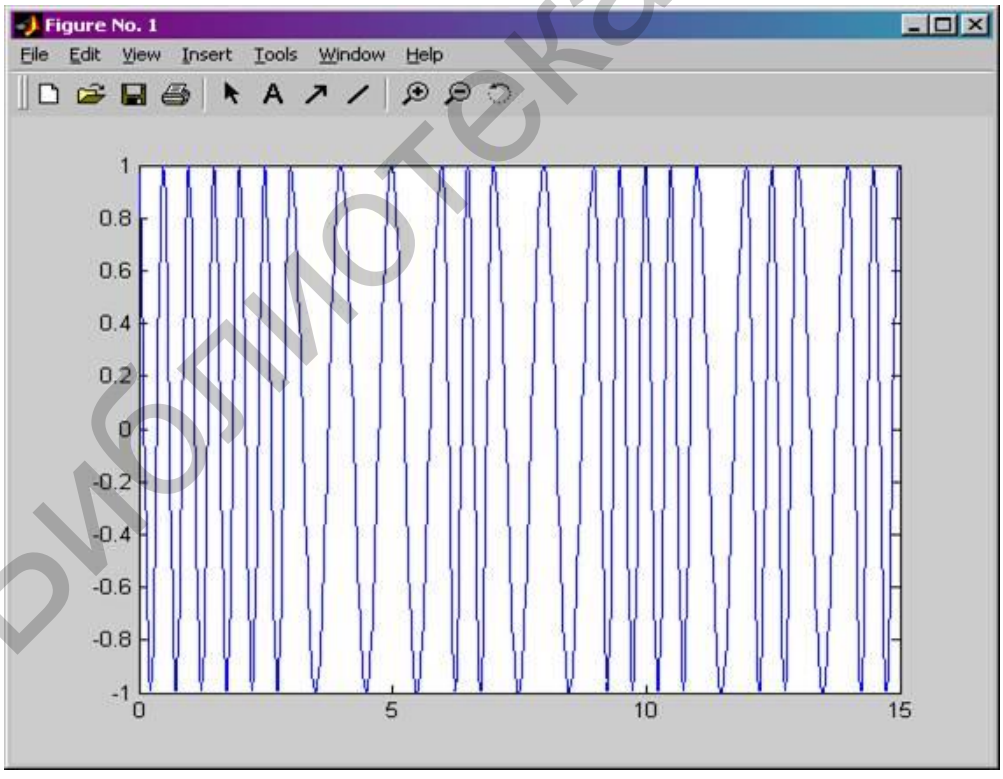


Рис. 16. Частотно-манипулированный сигнал во временной области для случая, когда манипулирующей функцией является [1 0 1 1 0 11 0 0 1 1 1 0 0 1 0 1 0 1]

На рис. 17 и 18 приведен спектр и АКФ частотно-манипулированного сигнала, когда манипулирующей функцией является 1 0 1 1 0 11 0 0 1 1 1 0 0 1 0 1 0 1 соответственно.

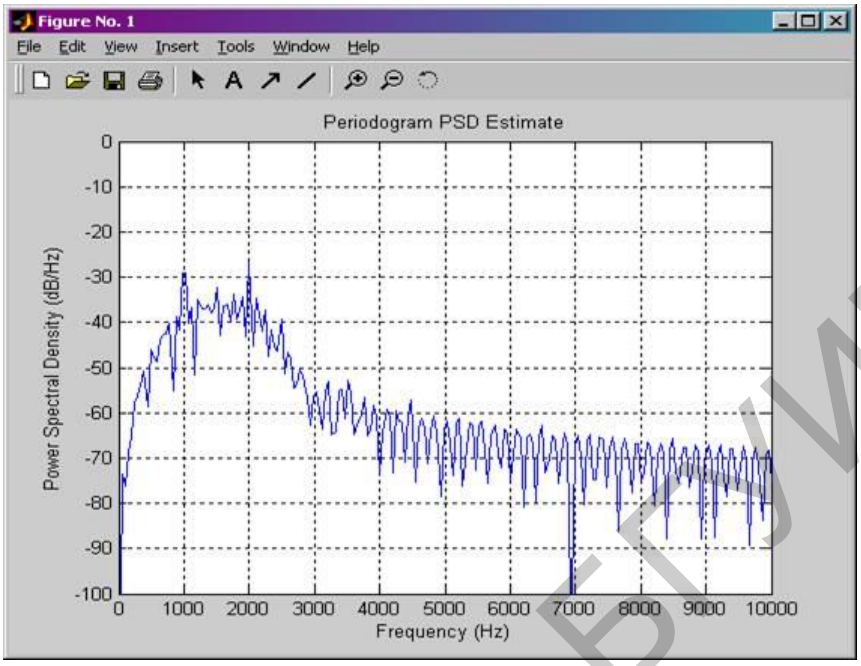


Рис. 17. Спектр мощности частотно-манипулированного сигнала для случая, когда манипулирующей функцией является [1 0 1 1 0 11 0 0 1 1 1 0 0 1 0 1 0 1]

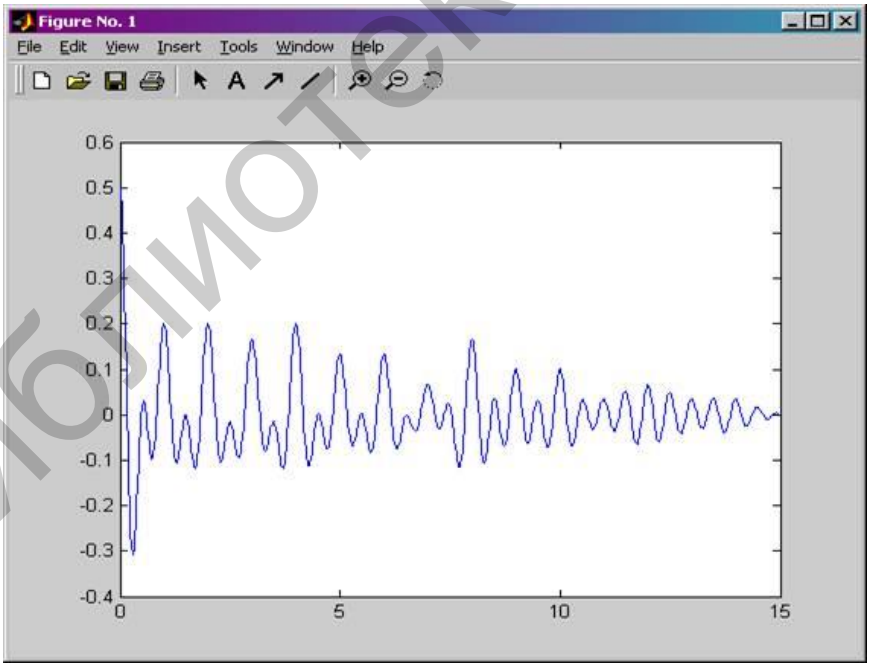


Рис. 18. АКФ частотно-манипулированного сигнала для случая, когда манипулирующей функцией является [1 0 1 1 0 11 0 0 1 1 1 0 0 1 0 1 0 1]

А теперь повторите вышеприведенный алгоритм еще раз, но уже для своего собственного цифрового сообщения.

2.3. Моделирование FSK-сигнала в пакете MATLAB-SIMULINK

Цифровой сигнал можно передать, используя средства системы связи. Например, FSK-модулированное колебание – это цифровой сигнал, который передается в эфир. Информационные биты или биты сообщений могут быть смоделированы, используя бинарный цифровой генератор Бернулли. Этот информационный бит подается на частотный модулятор, где биту 0 соответствует частота, модулированная одной частотой, и единичный бит модулируется другой частотой. Модулированный FSK-сигнал фильтруется с помощью гауссовского фильтра, и затем сигнал усиливается на требуемую величину, перед тем как его излучать в эфир.

Структурная схема этого процесса показана на рис. 19.

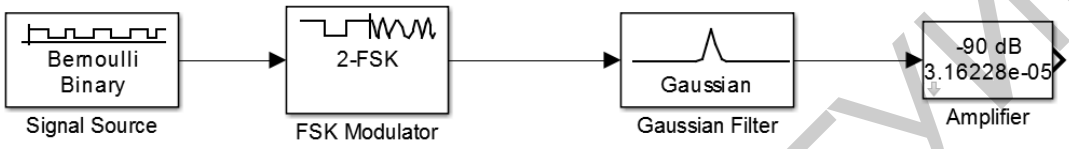


Рис. 19. Обработка сигнала перед подачей его в эфир

Каждый блок проектируется следующим образом.

Источник сигнала

Рассмотрим источник, который производит отсчеты с бинарными битами при частоте 1МГц (частота дискретизации). И отсчеты эти эмитируются в окне с 32 отсчета/окно. Пусть fs – частота дискретизации, равная 1МГц, и допустим на временное окно приходится число отсчетов в одном окне, которое представляет собой 32 отсчета/окно. Затем используем генератор Бернулли от источника связи, чтобы генерировать этот информационный сигнал. Установка параметров источника сигнала начинается с окна, показанного на рис. 20.

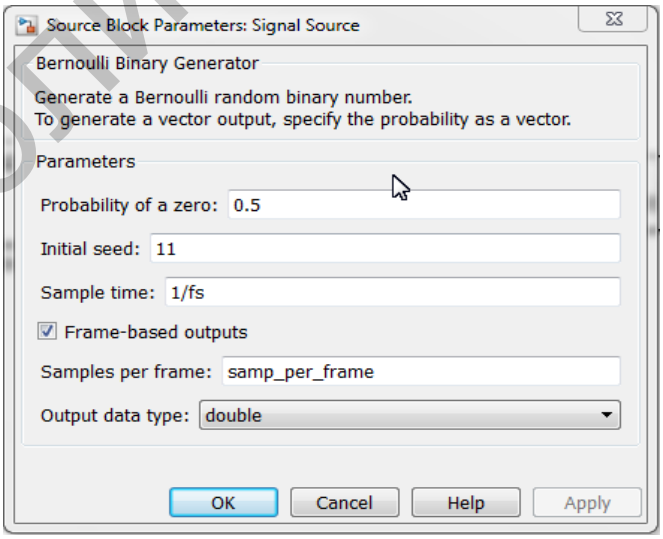


Рис. 20. Установка параметров источника сигнала

Время отсчета 1/fs равно 1мкс.

FSK-модуляция

Отсчеты, базирующиеся на временном окне, подаются на FSK-модулятор. FSK-модулятор располагается, как показано на рис. 21. Частотное разделение – это произведение частоты дискретизации и индекса модуляции. Пусть индекс модуляции обозначим как mod_index, который выбирается равным 0,33. Следующая вещь – это охарактеризовать отсчет/символ, который выбирается равным 16.

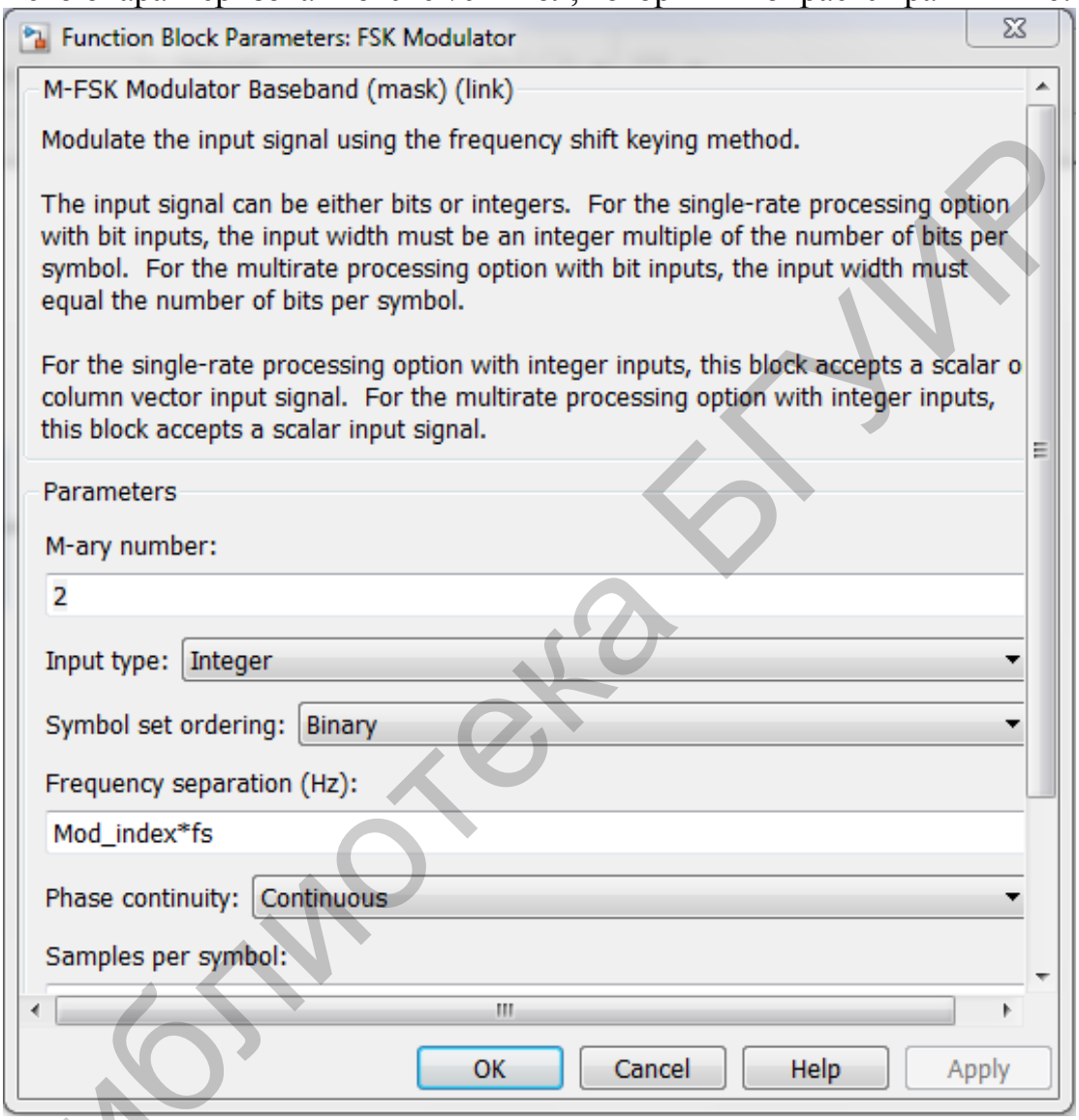


Рис. 21. Установка параметров FSK-модулятора

Гауссовский фильтр

Сигнал с манипулированной частотной модуляцией производится, когда он фильтруется гауссовским фильтром. Модулированный FSK-сигнал тоже фильтруется гауссовским фильтром. В гауссовском фильтре устанавливаем следующие значения – Input samples per symbols(N)= samp_per_sym = 16 и BT product, BT = 0.5, как это показано на рис. 22.

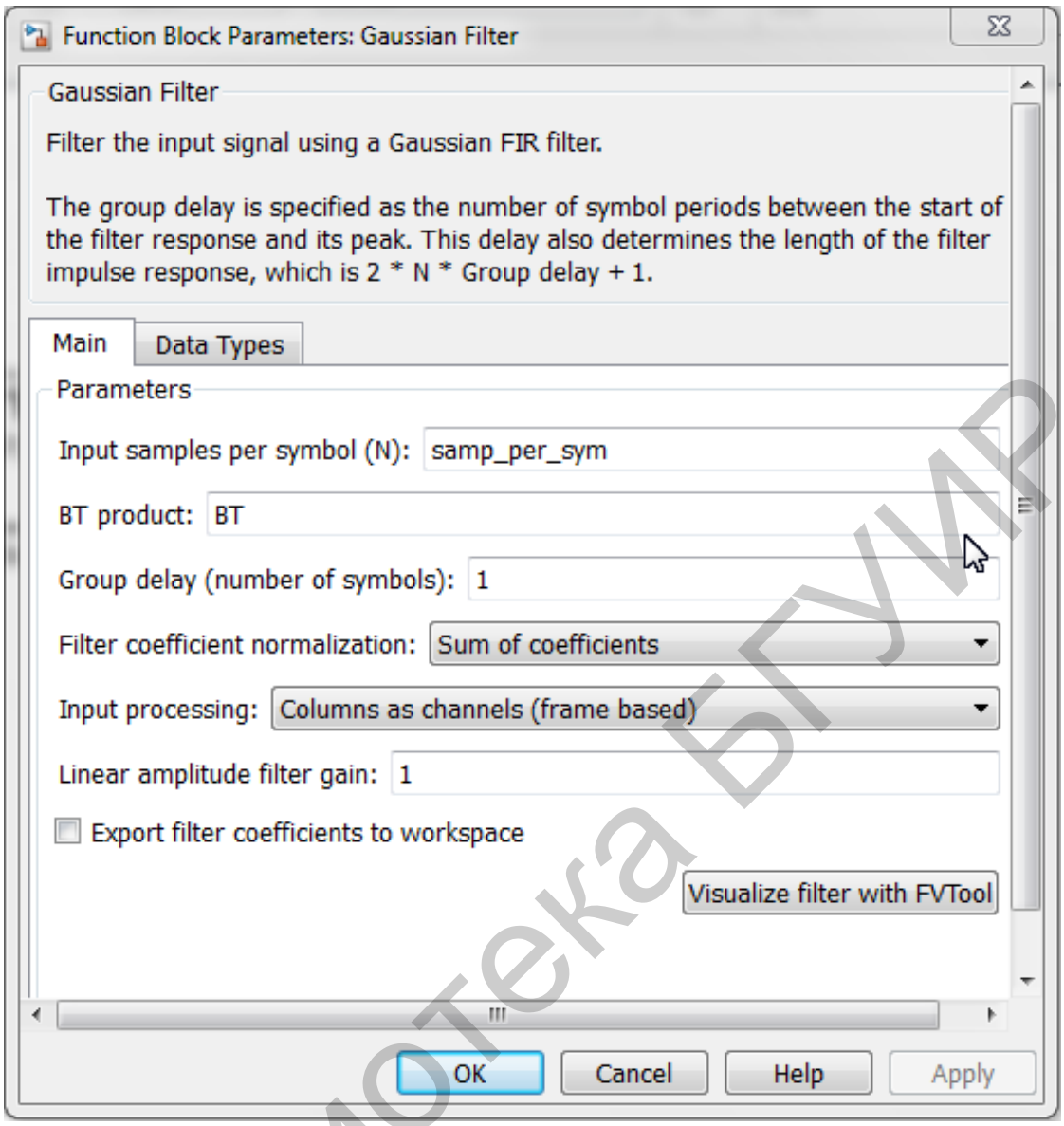


Рис. 22. Установка параметров для гауссовского фильтра

Усилитель

Фильтрованный FSK-сигнал усиливается частью простого усиления, как показано на рис. 19, который имеет усиление порядка 90 дБ.

Моделирование

Можно моделировать работу системы, структурная схема которой приведена на рис. 19. Например, на рис. 23 показаны бинарные биты и соответствующий FSK-сигнал.

Кроме того, можно наблюдать спектр модулированного сигнала до и после фильтрования. На рис. 24 показан спектр сигнала перед фильтрацией. После фильтрации спектр FSK-модулированного сигнала показан на рис 25.

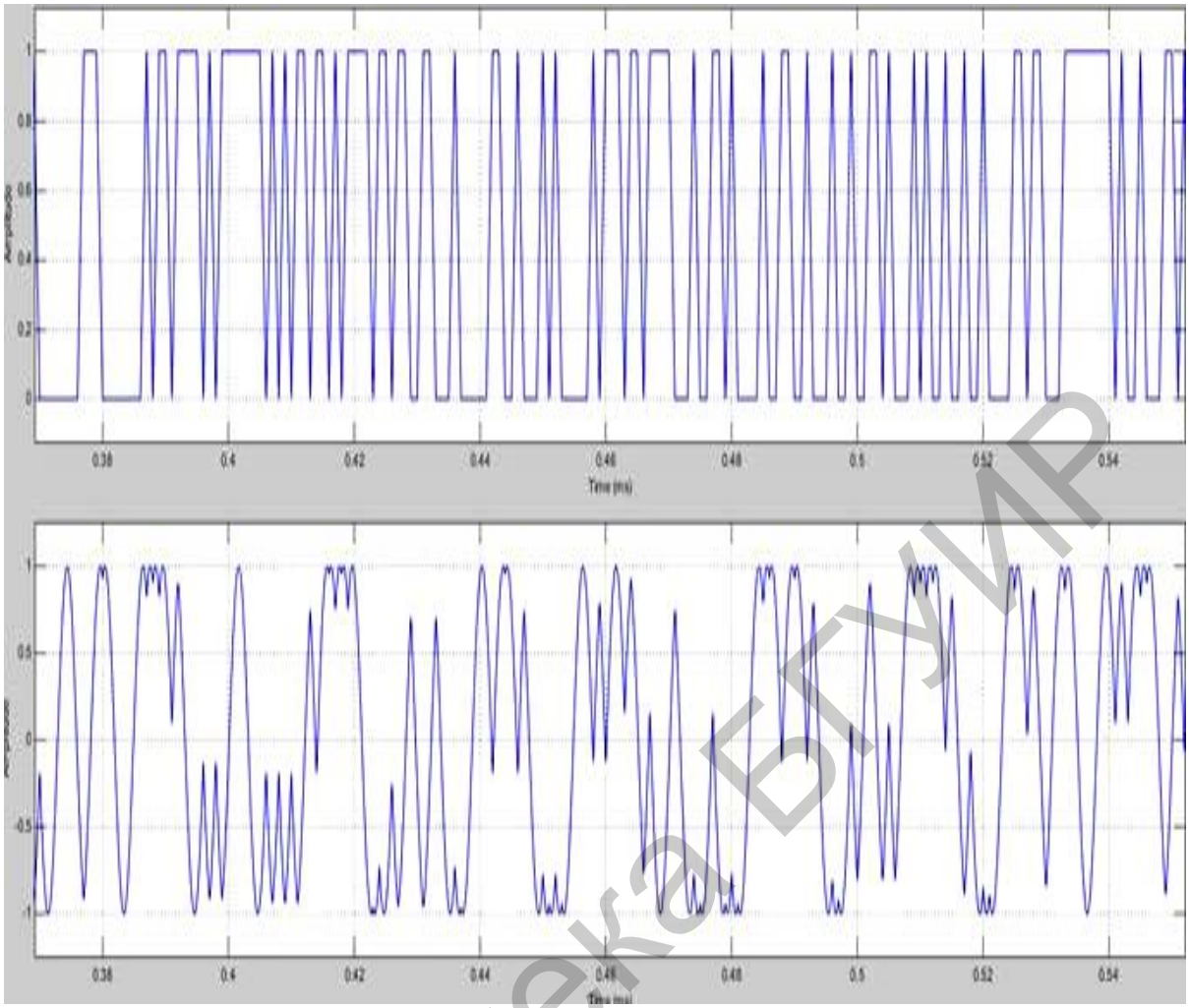


Рис. 23. Бинарные биты и соответствующий сигнал

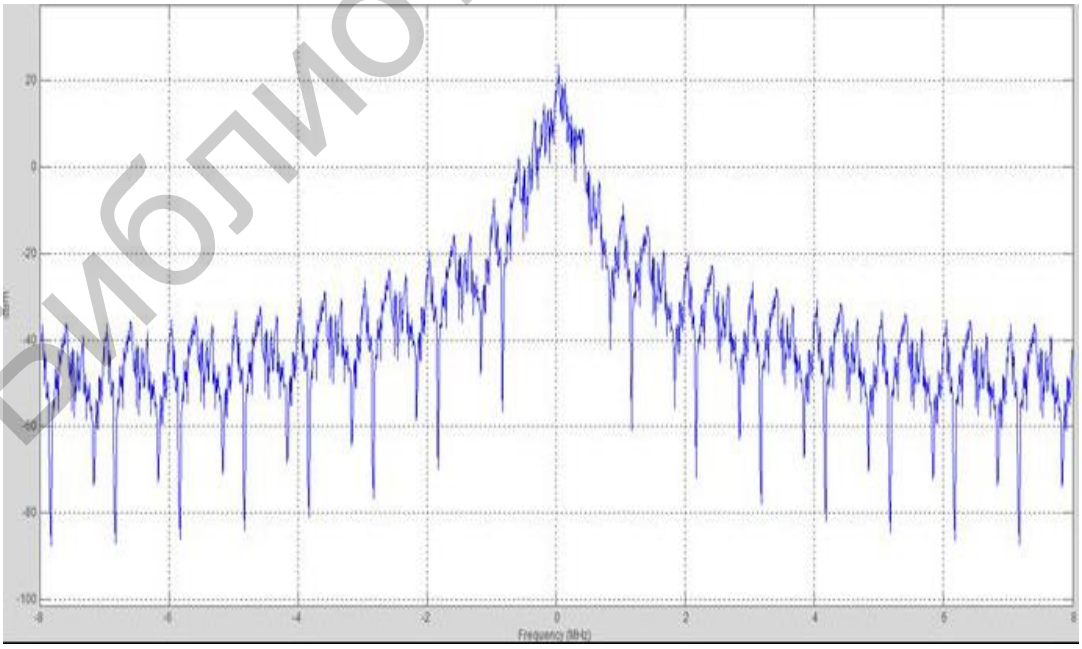


Рис. 24. Спектр сигнала перед фильтрацией

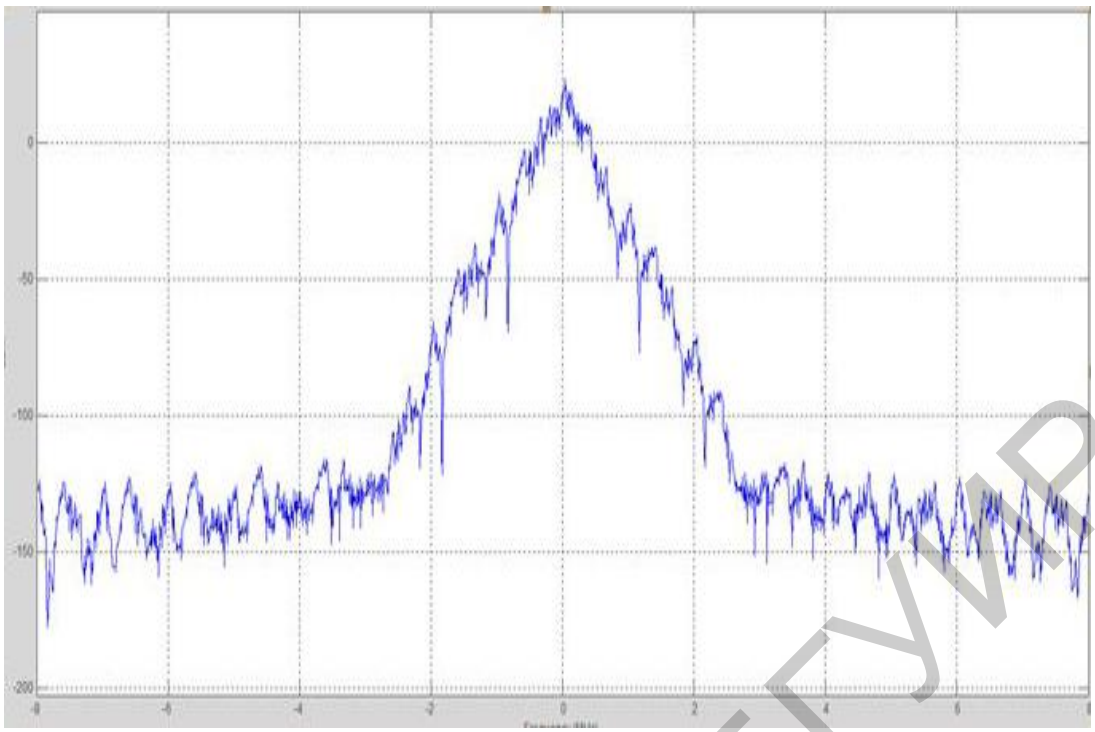


Рис. 25. Спектр сигнала после фильтрации

Наконец усиленный сигнал показан на рис. 26.

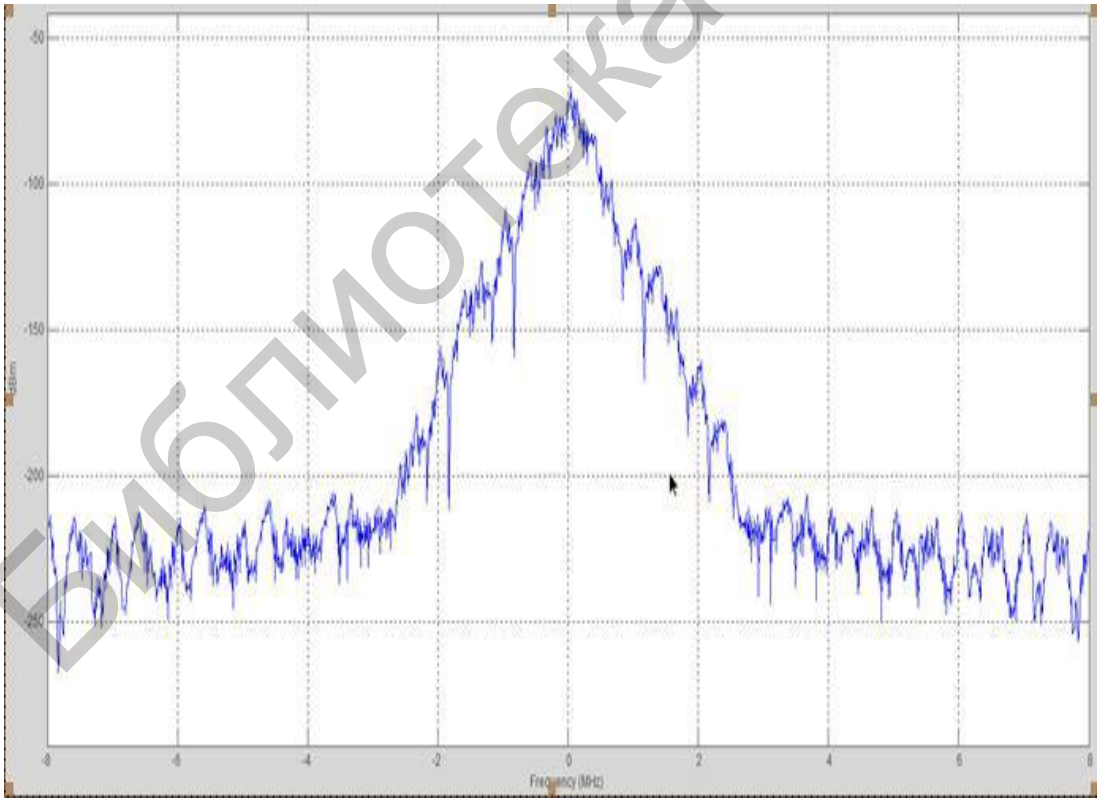


Рис. 26. Сигнал после усиления

Заметим, амплитуда увеличивается в дБ по оси y.

Соберите схему, приведенную на рис. 19, и сравните результаты ее работы с приведенными.

Для более глубокого изучения процессов частотной манипуляции рассмотрим моделирование М-ичной частотной манипуляции в системе SIMULINK.

1. Войдем в программу SIMULINK.

2. Создадим новую модель для моделирования процессов в системе с частотной М-ичной манипуляцией, для чего выберем опции **File** -> **New** -> **Model** в браузере **Simulink Library Browser**, общий вид которого показан на рис. 27

3. Перейдем к **Communications Blockset** -> **Comm Sources** -> **Random Data Sources**, перетащим модуль **Random Integer Generator** в модельное окно программы. Двойным щелчком мыши откроем модуль **Random Integer Generator** и установим следующие его параметры, как это видно из рис. 28:

- M−ary number to 2
- Initial seed to 37
- Sample time to 0.1
- Output Data Type to double

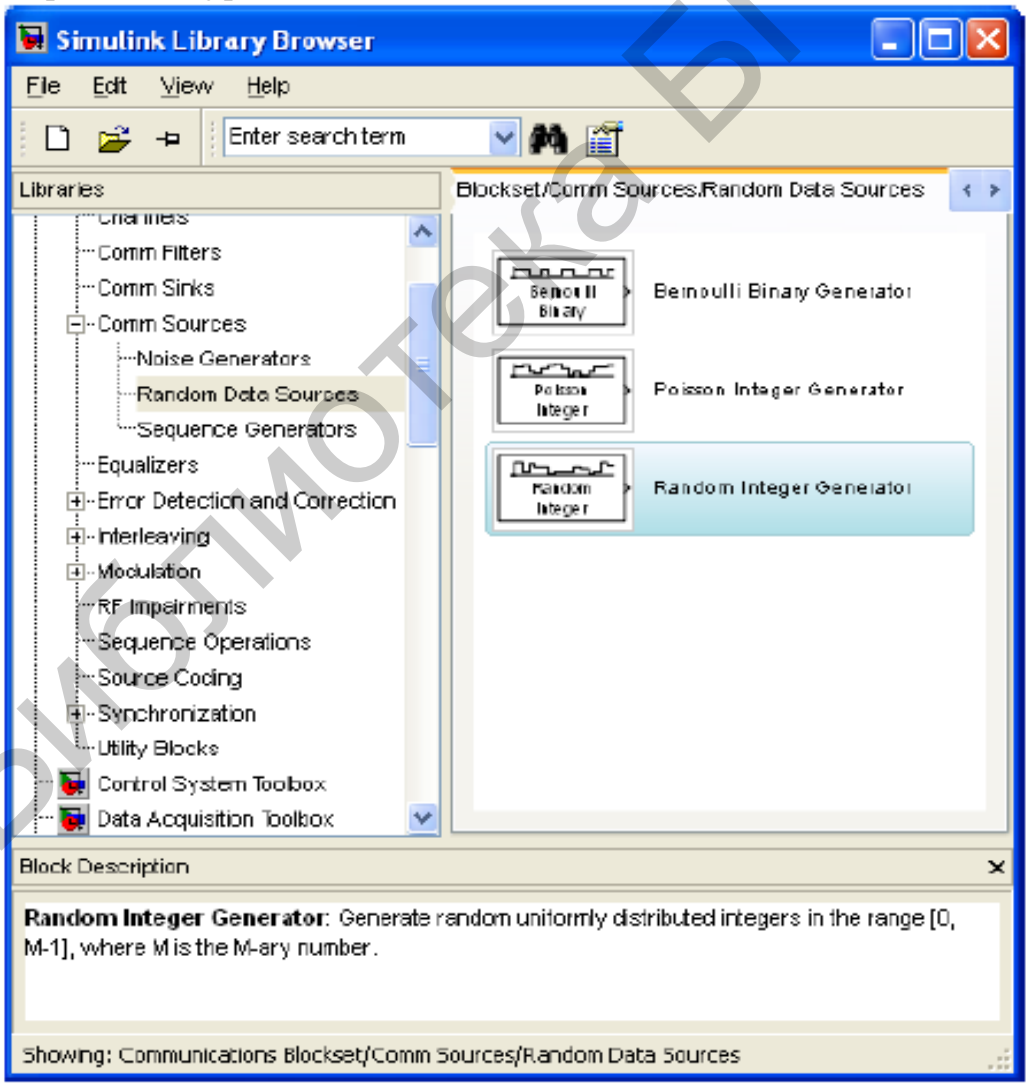


Рис. 27. Общий вид браузера **Simulink Library Browser**

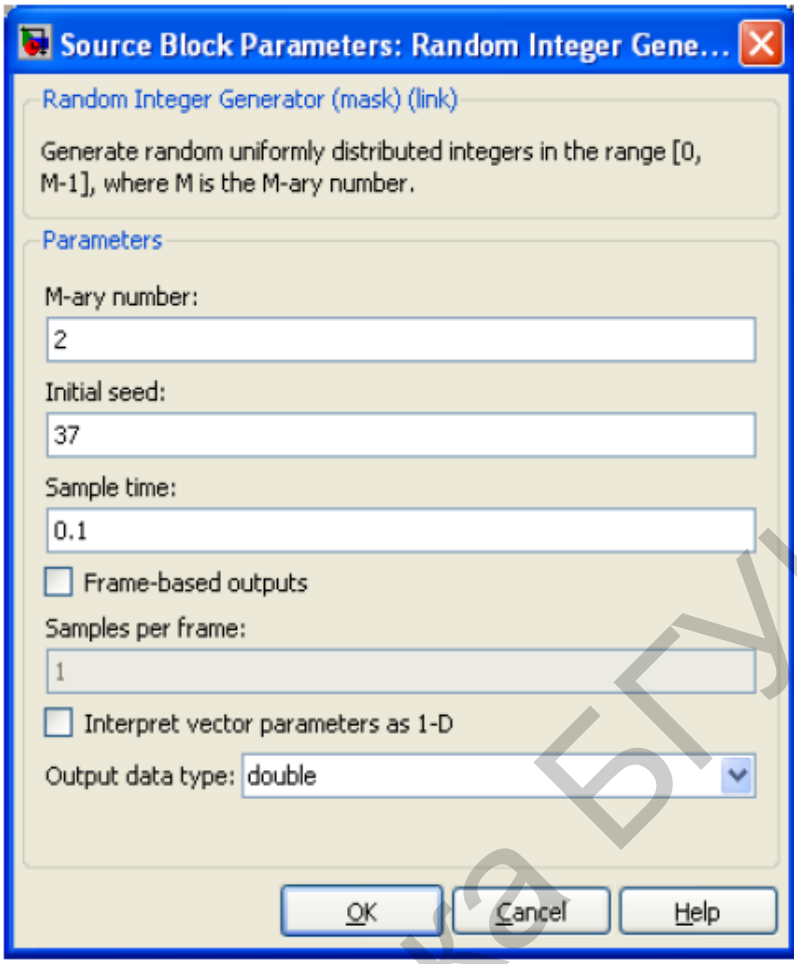


Рис. 28. Общий вид панели для установки параметров модуля **Random Integer Generator**

4. Перейдем теперь к **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** -> **FM**, перетащим модуль **M-FSK Modulator Baseband** в модельное окно программы, как это видно из рис. 29, и, открыв двойным щелчком мыши модуль **M-FSK Modulator Baseband**, установим следующие его параметры, как это показано на рис. 30:

- M−ary number to 2
- Input type to Integer
- Symbol set ordering to Binary
- Frequency separation (Hz) to 6
- Phase continuity to Continuous
- Samples per symbol to 100
- Output Data Type to double.

5. Перейдем к **Communications Blockset** -> **Channels**, перетащим модуль **AWGN Channel** в модельное окно программы и, открыв двойным щелчком мыши модуль **AWGN Channel**, установим следующие его параметры:

= Initial seed to 37

- Mode to Signal-to-noise ratio (Eb/No)

- Eb/No (dB) to 10
- Number of bits per symbol to 4
- Input signal power (watts) to 1
- Symbol period(s) to 0.1.

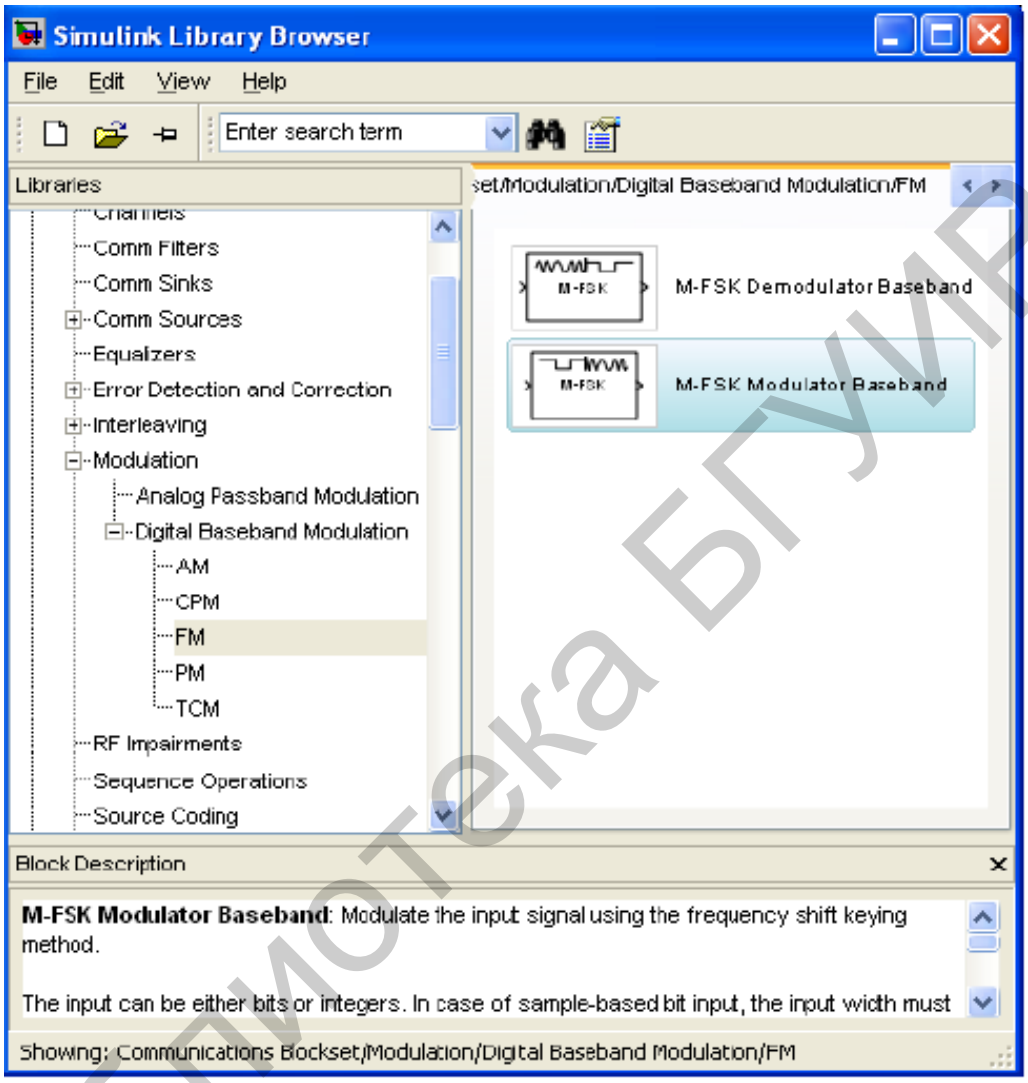


Рис. 29. Общий вид браузера **Simulink Library Browser** при выборе модуля **M-FSK Modulator Baseband**

6. Перейдем к **Communications Blockset** -> **Modulation** -> **Digital Baseband Modulation** -> **FM**, перетащим модуль **M-FSK Demodulator Baseband** в модельное окно программы и, открыв двойным щелчком мыши модуль **M-FSK Demodulator Baseband**, установим следующие его параметры:

- M−ary number to 2
- Output type to Integer
- Symbol set ordering to Binary
- Frequency separation (Hz) to 6
- Samples per symbol to 100
- Output Data Type to double.

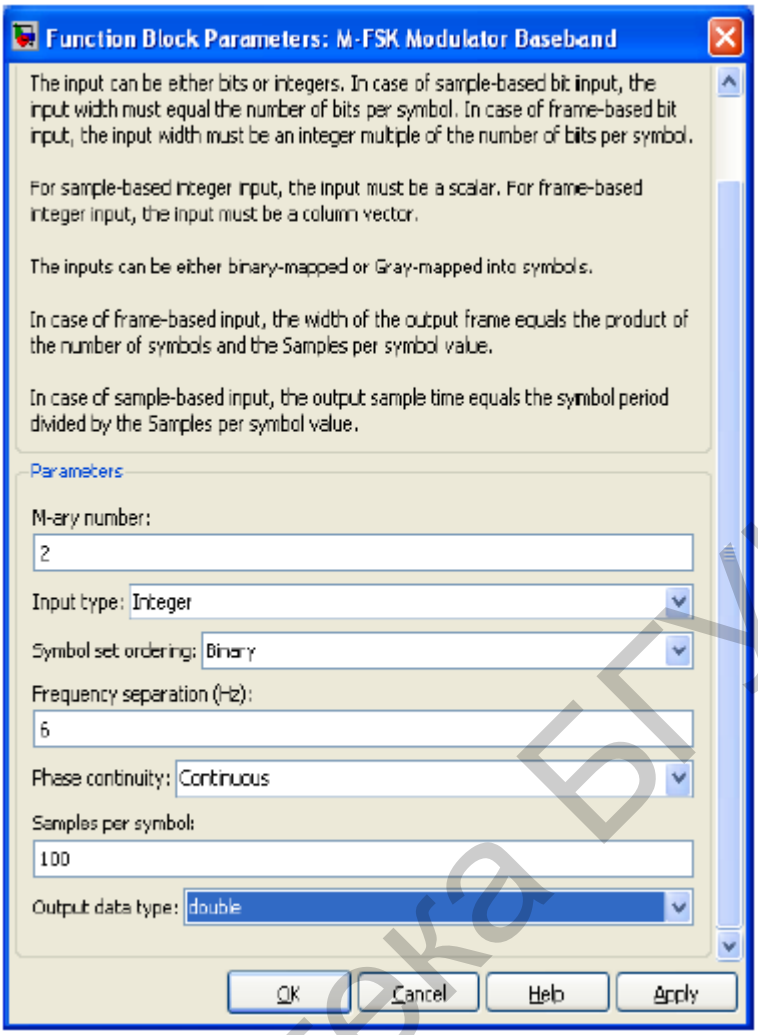


Рис. 30. Общий вид панели для установки параметров модуля **M-FSK Modulator Baseband**

7. Перейдем к **Communications Blockset** -> **Comm Sinks**, перетащим модуль **Error Rate Calculation** в модельное окно программы.

8. Перейдем к **Simulink** -> **Sinks**, перетащим модуль **Display** в модельное окго программы, сделав его достаточно большим для подключения трех входов.

9. Перейдем к **Communications Blockset** -> **Comm Sinks**, перетащим два модуля **Discrete-Time Scatter Plot Scope** в модельное окно программы и, дважды щелкнув мышью на этом модуле, откроем его и установим следующие параметры.

: **Plotting Properties**

- Samples per symbol to 100
- Offset (samples) to 1

= Points displayed to 400.

- New points per display to 100

Axes Properties

- X-axis minimum to -1.25
- X-axis maximum to 1.25

- Y-axis minimum to -1.25

- Y-axis maximum to 1.25.

10. Перейдем к **Simulink** -> **Math Operations**, перетащим модуль **Complex to Real−Imag** в модельное окно программы.

11. Перейдем к **Simulink** -> **Sinks**, перетащим модуль **XY Graph** в модельное окно программы.

12. Перейдем к **Simulink** -> **Sinks**, перетащим два модуля **Scope** в модельное окно программы.

13. Перейдем теперь к **Simulink Extras** -> **Additional Sinks**, перетащим два модуля **Power Spectral Density** в модельное окно программы.

14. Соединим все модули, как показано на рис. 31.

15. Установим следующие параметры моделирования **Simulation** -> **Configuration Parameters**, как это показано на рис. 32:

- Start time to 0.0

- Stop time to 10.0

- Type to Variable-step

- Solver to discrete (no continuous states)

- Max. step size to auto.

16. Запустим программу на выполнение, сохраним все диаграммы и значения измеренных величин, осмыслим их и сделаем выводы, которые следует отразить в отчете о проделанной лабораторной работе.

17. Вернемся снова к моделированию, изменив М-ичное число в модулях **Random Integer Generator**, **M-FSK Modulator Baseband** и **M-FSK Demodulator Baseband**, выбрав его равным 4, 8, 16, 32, и 64.

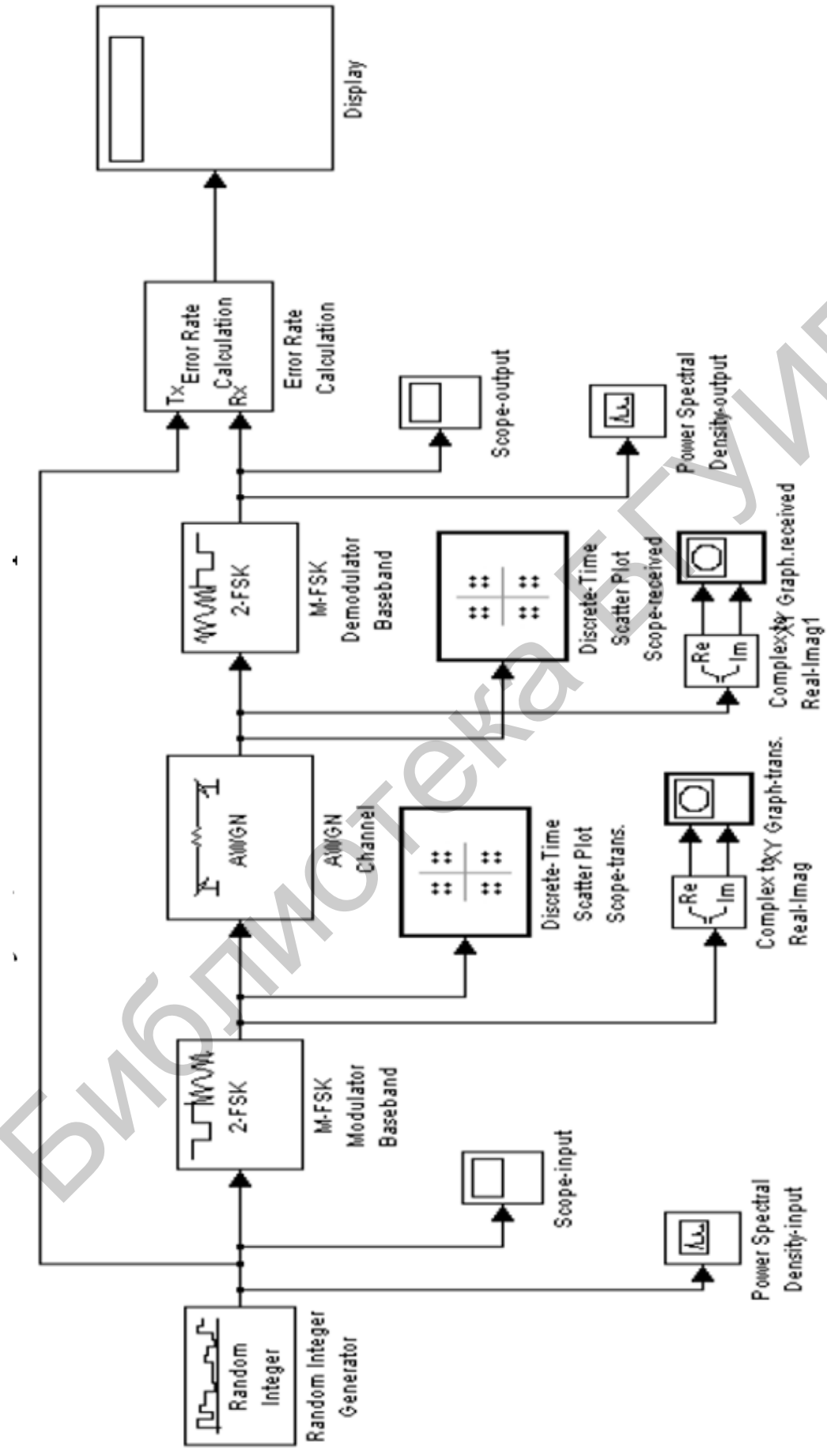


Рис. 31. Схема моделирования М-ичной частотной манипуляции

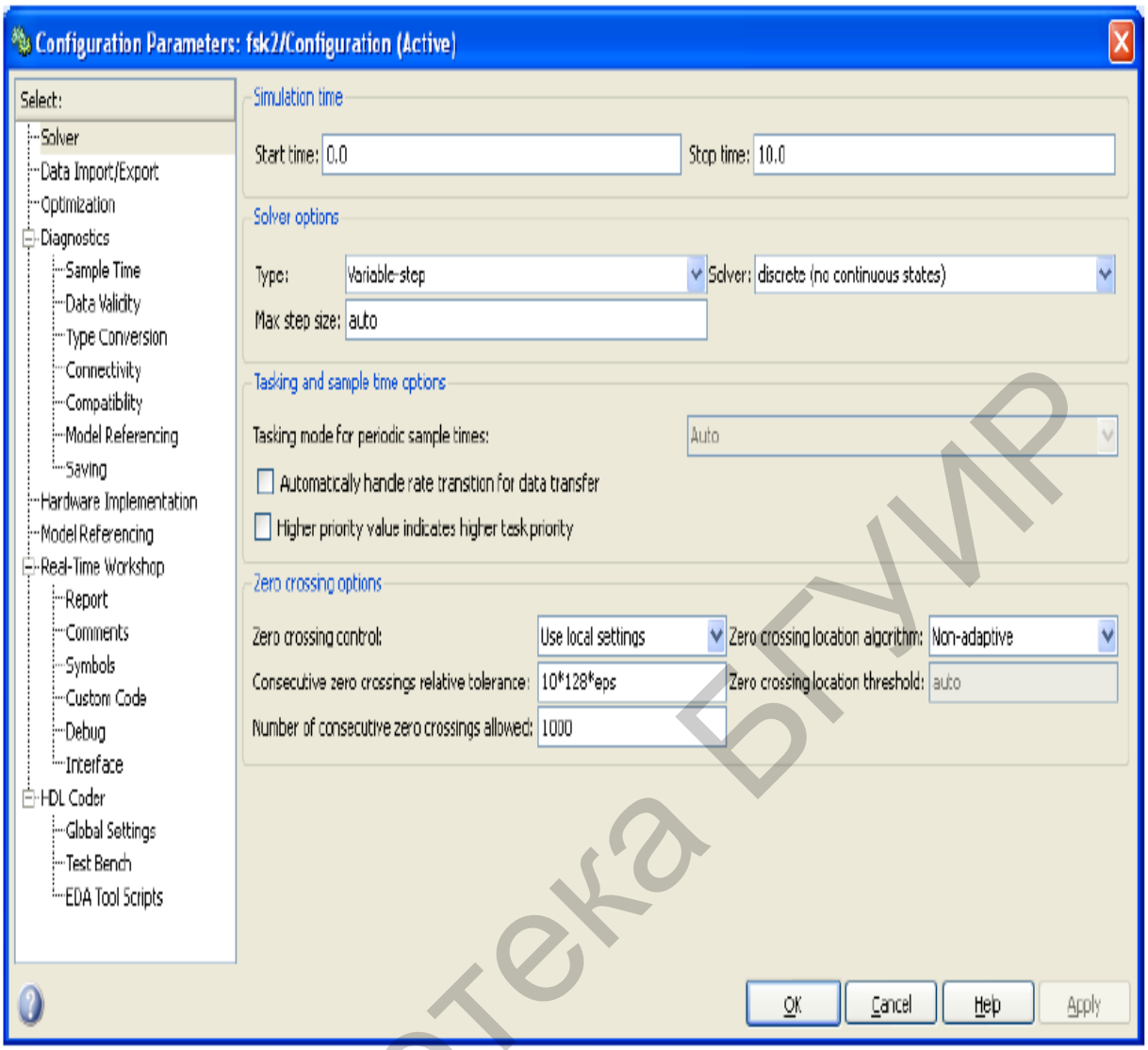


Рис. 32. Выбор параметров моделирования

3. Контрольные вопросы

1. Назначение генератора Бернулли в лабораторной работе.
2. Назначение гауссовского фильтра.
3. Понятие частотно-манипулированного сигнала.
4. Запишите расчетную формулу для частоты девиации FSK-сигнала.
5. Понятие бинарной частотной манипуляции.
6. Основные характеристики спектра FSK-сигнала.
7. Последовательность построения частотно-манипулированного сигнала.
8. Понятие частотного разделения при бинарной частотной манипуляции.
9. Физический смысл параметра K, использующегося в данной лабораторной работе.
10. В чем состоит моделирование автокорреляционной функции частотно-манипулированного сигнала.

Лабораторная работа №6
ИЗУЧЕНИЕ СЛОЖНЫХ МЕТОДОВ ФОРМИРОВАНИЯ ФАЗО- И ЧАСТОТНО-МАНИПУЛИРОВАННЫХ КОЛЕБАНИЙ С ИСПОЛЬЗОВАНИЕМ ПАКЕТА MATLAB

1. Краткие теоретические сведения

В этой работе будет использовано объектно-ориентированное программирование пакета MATLAB для построения объектов и методов, используемых этой системой для анализа и изучения более сложных методов формирования фазо- и частотно-манипулированных колебаний.

1.1. Моделирование модуляции сигналов с помощью разных методов объекта modem

При выполнении модуляции изменяется амплитуда, несущая частота или начальная фаза несущего колебания в соответствии с изменениями передаваемого по каналу связи сигналом сообщения.

Модулирующая функция выполняет процесс модуляции, используя специфические методы модуляции.

Синтаксис модулирующей функции имеет вид

y = modulate(x, fc, fs, 'method', opt),

где x – сигнал сообщения; fc – несущая частота; fs – частота дискретизации; method – флаг для нужного метода модуляции, некоторый дополнительный аргумент, необходимый для выполнения выбранного метода. (Не все методы модуляции требуют наличия этого аргументе).

В приведенной ниже табл. 1 содержатся соответствия используемых методов и видов модуляции, где они применяются.

Таблица 1

Соответствие методов и видов модуляции, где они применяются

Метод модуляции	Вид модуляции, где он используется
amdsb-sc или am	Амплитудная модуляция с двумя боковыми полосами с подавленной несущей частотой
amdsb-tc	Амплитудная модуляция с двумя боковыми полосами с передаваемой несущей частотой
Amssb	Амплитудная модуляция с одной боковой полосой
Fm	Частотная модуляция
Pm	Фазовая модуляция
Ppm	Модуляция
Pwm	Широтно-импульсная модуляция
Qam	Квадратурная амплитудная модуляция

Если входной сигнал x представлен массивом, а не вектором, метод modulate модулирует тогда каждый столбец массива. Чтобы получить соответствие значений вектора временным моментам, которое используется при расчете модулированного сигнала, в его описание вводится еще один параметр t:

[y, t] = modulate(x, fc, fs, ‘method’, opt);

1.2. Моделирование демодуляции сигналов с помощью разных методов объекта modem

Функция demod выполняет демодуляцию, т. е. она получает исходный сигнал сообщения из модулированного сигнала. Синтаксис этой функции имеет следующий вид:

x = demod(y, fc, fs, 'method', opt).

Функция demod использует любой метод, примененный для модуляции, но синтаксис для квадратурной модуляции требует двух выходных параметров:

[X1,X2] = demod(y, fc, fs, 'qam').

Если входной сигнал *y* представляет собой массив, то функция demod демодулирует все столбцы.

Пример 1. Промодулируем и демодулируем сигнал, который представляет собой синусоиду частотой 50 Гц, дискретизированной со скоростью 1000 Гц. Имеем

```
>>t = (0:1/1000:2);
>>x = sin(2*pi*50*t);
```

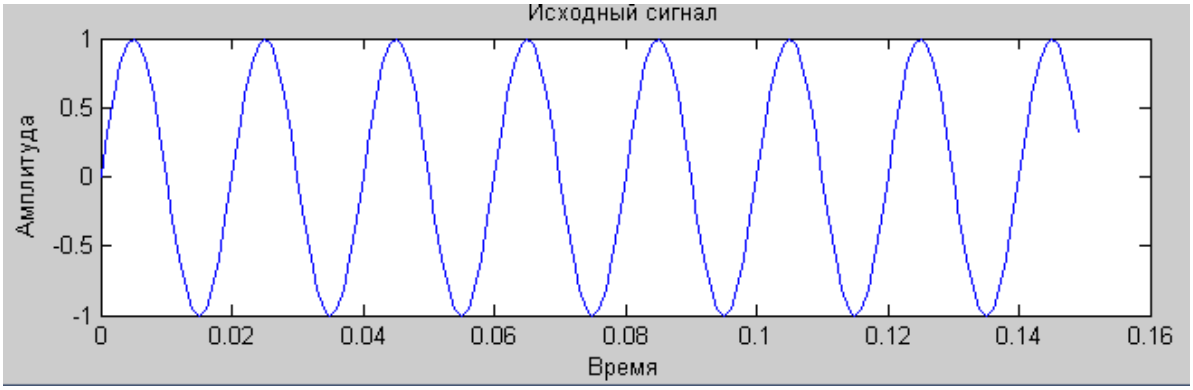
Если несущая частота выбрана равной 200 Гц, тогда модулирующий и модулированный сигналы имеют вид

```
>>y = modulate(x,200,1000,'am');
>>z = demod(y,200,1000,'am');
```

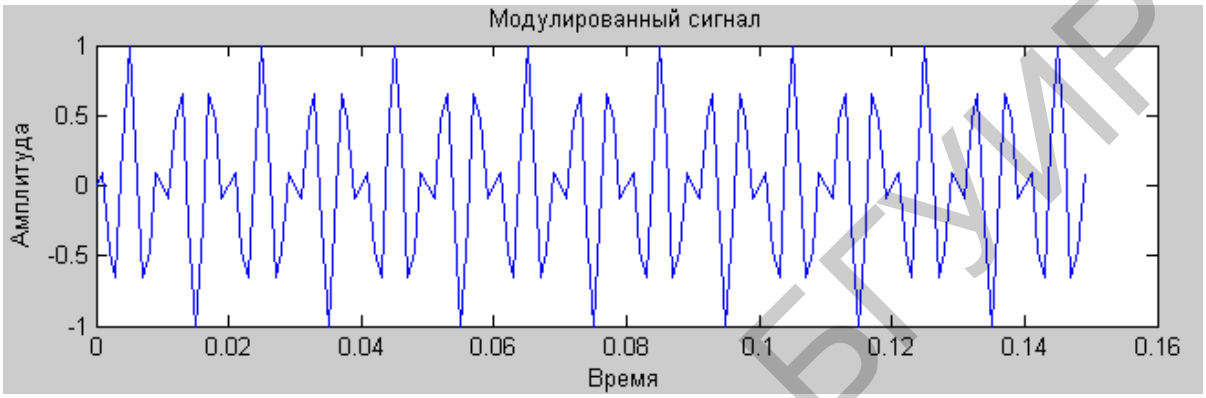
Запишем коды MATLAB для вывода графиков этих трех сигналов:

```
>>figure; plot(t(1:150),x(1:150)); title('Исходный сигнал);
>> ylabel('Амплитуда');
>> xlabel('Время');
>>figure; plot(t(1:150),y(1:150)); title(‘Модулированный сигнал’);
>> ylabel('Амплитуда');
>> xlabel('Время');
>>figure; plot(t(1:150),z(1:150)); title(‘Демодулированный сигнал’);
>> ylabel('Амплитуда');
>> xlabel('Время');
```

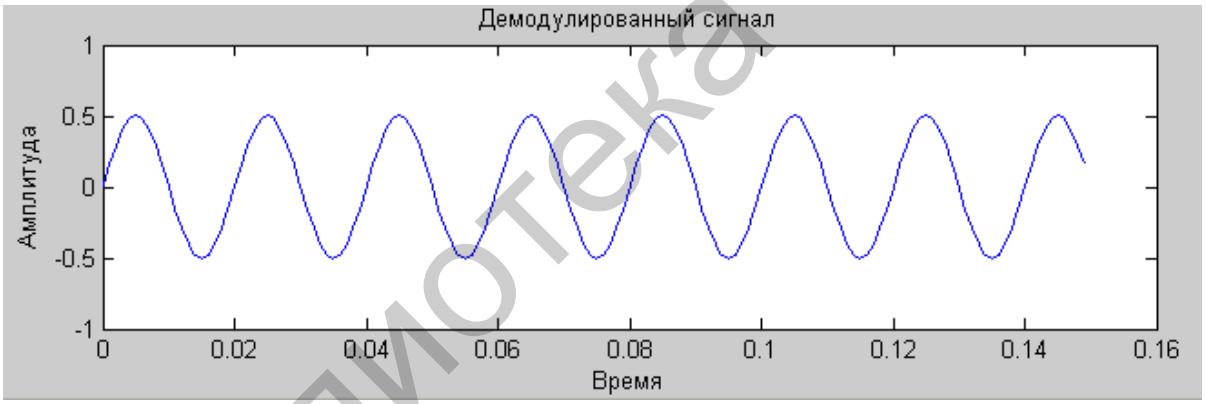
Представим на рис. 1 диаграммы исходного, модулированного и демодулированного сигналов.



а



б



в

Рис. 1. Графики исходного (а), модулированного (б)
и демодулированного (в) сигналов

Из рис. 1 видно, что демодулированный сигнал затухает, так как процесс демодуляции включает выполнение перемножения и низкочастотной фильтрации. Процедура перемножения создает компоненту с нулевой средней частотой и компоненту с частотой, вдвое превышающей несущую частоту. Фильтрация устраняет верхнюю частоту компоненты сигнала, что приводит к ослаблению выходного сигнала демодулятора.

1.3. Использование объекта modem

Вообще модуляция сигналов требует выполнения функций, таких как fskmod или ssbmod. Для DPSK, General QAM, MSK, OQPSK, PAM, PSK и QAM

можно модулировать сигналы, используя объекты типа modem. Здесь покажем, как можно использовать эти объекты.

Объект modem – это тип переменной системы MATLAB, что содержит информацию об алгоритме модуляции, такую как имя класса, M-арное число, и форму представления созвездия. Объект может выполнить свою работу, используя специфические методы для выполнения определенных задач.

1.3.1. Конструирование объекта modem

Для конструирования объектов модулятора и демодулятора используются функции (конструкторы), показанные в табл. 2.

Таблица 2

Типы модуляции и конструкторы

Тип модуляции	Конструкторы
DPSK	modem.dpskmod и modem.dpskdemod
General QAM	modem.genqammod и modem.genqamdemod
MSK	modem.mskmod и modem.mskdemod
OQPSK	modem.oqpskmod и modem.oqpskdemod
PAM	modem.pammod и modem.pamdemod
PSK	modem.pskmod и modem.pskdemod
QAM	modem.qammod и modem.qamdemod

1.3.2. Управление свойствами объекта

Чтобы ознакомиться со свойствами объекта, используется метод disp, как это показано в следующем примере.

Пример 2.

```
>>h=modem.pskmod; % Конструируем объект модулятора
>>h.disp  % Выводим на дисплей свойства этого  объекта
                    Type:    'PSK Modulator'
                       M:    2
             PhaseOffset:    0
           Constellation:    [1 -1.0000 + 0.0000i]
             SymbolOrder:    'Binary'
          SymbolMapping:    [0 1]
               InputType:    'Integer'
```

Можно непосредственно назначить значения свойств этого объекта следующим образом:

```
>>h=modem.pskmod(8);    % конструируем объект модулятора PSK
>>h.symbolorder='gray'; % устанавливаем свойство 'symbolorder' объекта, которое называется 'gray'.
```

В результате выполнения этих команд получается объект со свойствами:

```
>> h.disp
              Type:    'PSK Modulator'
                 M:    8
       PhaseOffset:    0
     Constellation:    [1x8 double]
       SymbolOrder:    'Gray'
    SymbolMapping:    [0 1 3 2 6 7 5 4]
         InputType:    'Integer'
```

Свойства можно установить также во время конструирования объекта.

1.3.3. Копирование объекта modem

Синтаксис выполнения этой операции

>>syntax h = copy(refobj)

создает новый образец объекта *h* того же самого типа, что и объект *refobj* и копирует все свойства объекта refobj в объект *h*. Назначение другой переменной объекту копирует только указатель и поэтому не создает независимой ее копии. Таким образом, в предыдущем примере если установлено *a = h*, тогда *a* указывает на тот же самый объект *h* и любое изменение делает h также отраженным в *a*.

1.3.4. Восстановление объекта modem

Объекты modem для MSK, OQPSK и DPSK, т. е. только эти объекты, которые имеют память, имеют метод reset, который восстанавливает внутреннюю структуру объекта. Он допускает, что число каналов входного сигнала для методов модуляции и демодуляции равно единице, т. е. вход представляет собой вектор-столбец.

Команда reset(h,nchan) представляет внутреннее состояние объекта h, допуская число каналов nchan, где вход модулятора является матрицей с nchan столбцами. Если метод modulate или demodulate называется с числом каналов, отличающимся от nchan, объект автоматически восстанавливается с правильным числом каналов. На следующем примере покажем, как это происходит

Пример 3. Демонстрация использования команды reset. Имеем

```
>>h = modem.mskmod; %  создаем объект со свойствами по умолчанию
>>x = randint(100, 1, 2); % генерируем входные биты
>>y = modulate(h, x); % модулируем сигналом сообщения  x
>>x = randint(100, 1, 2); % генерируем новые входные биты
>>reset(h); % восстанавливаем модулятор
>>y = modulate(h, x); %модулируем x с тем же самым исходным состоянием, что и в первом случае
```

1.3.5. Модулирование сигнала

Основная процедура для модулирования сигнала с DPSK, MSK, OQPSK, PAM, PSK, QAM или general QAM включает следующие этапы:

1. Конструируем объект модулятора, как это описано в п. 1.3.1, в зависимости от выбранного типа модуляции.

2. Согласуем свойства объекта модулятора, если необходимо, чтобы удовлетворить требования тех задач, которые должен решать модулятор. Например, можно изменить фазу offset или порядок символов.

3. Модулировать сигнал, используя метод modulate объекта модулятора, как описано в следующем разделе.

Метод модуляции объекта modem

Объект модулятора имеет метод modulate, что используется для модуляции сигналов. Синтаксис выполнения этих операций сводится к следующему:

>> y = modulate(h, x),

где *h* – указатель на модулятор объекта и x – сигнал сообщения. Этот синтаксис обеспечивает выходной сигнал *y*.

Сигнал *x* может быть многоканальным сигналом. Столбцы *x* рассматриваются как индивидуальные каналы, в то время как строки соответствуют временным отсчетам.

Когда входные биты образуют символ, первый бит представляет собой старший разряд формируемого бинарного числа.

При выполнении команды

>>h.inputtype = ‘bit'

(т. е. сигнал x представляет собой бинарную последовательность), nBits последовательных элементов в каждом канале или столбце представляют символы, где nBits = log2(h.M). Число элементов в каждом канале должно быть целым, умноженным на nBits, и элементы x должны быть равны 0 или 1. Для размера входных сигналов *x*, рассчитывается размер сигнала *y* на выходе.

При выполнении команды

>>h.inputtype = ‘integer',

(т. е. сигнал представляет собой входные символы) элементы *x* должны быть в пределах [0, h.M-1]. Для размера входных сигналов *x* рассчитывается размер сигнала *y* на выходе.

1.3.6. Демодулирование сигнала

Основная процедура для модулирования сигнала с DPSK, MSK, OQPSK, PAM, PSK, QAM или general QAM включает следующие этапы:

1. Конструируем объект демодулятора, как это описано в п. 1.3.1, в зависимости от выбранного типа модуляции.

2. Согласуем свойства объекта демодулятора, если необходимо, чтобы удовлетворить требования тех задач, которые должен решать демодулятор. Например, можно изменить фазу offset или порядок символов.

3. Демодулировать сигнал, используя метод demodulate объекта демодулятора, как описано в следующем разделе.

Метод демодуляции объекта modem

Объект демодулятора имеет метод demodulate, что используется для демодуляции сигналов. Синтаксис выполнения этих операций сводится к следующему:

>> y = demodulate(h, x),

где *h* – указатель на модулятор объекта и x – сигнал сообщения. Этот синтаксис обеспечивает выходной сигнал *y*. Этот синтаксис обрабатывает бинарные слова (биты) или символы (целые числа), составляющие содержание сигнала сообщения *x*, демодулятором PSK или QAM объекта и выходом основного сигнала *y*. Сигнал *x* может быть многоканальным сигналом. Столбцы сигнала *x* рассматриваются как индивидуальные каналы, в то время как строки соответствуют временным отсчетам.

Свойства DecisionType демодулятора объекта должны быть установлены в зависимости от того, нужно ли работать с мягким или жестким разрешением

(приближенное LLR или LLR. Для реализации мягкого разрешения свойство OutputType демодулятора объекта должно быть установлено в 'bit'.

При выполнении команды

>>h.inputtype = ‘bit'

размер выхода y рассчитывается для размера входа x, где nBits = log2(h.M).

Для

>> h.outputtype = ‘integer'

размер выхода рассчитывается для размера входа *x*.

Пример 3. Составим коды, чтобы продемонстрировать основные этапы процедуры выполнения модуляции и демодуляции. Вот эти коды.

```
>>x = randint(10,1,8);   % создаем источник сигнала
>>h = modem.qammod(8)    % создаем объект модулятора и выводим его свойства на дисплей
>>y = modulate(h,x);     % модулируем сигнал y сигналом x
>>g = modem.qamdemod(h)  % Создаем объект демодулятора из объекта modem.qammod и выводим на дисплей его свойства
>>z = demodulate(g,y);   % Демодулируем сигнал y.
```

Получаем

```
h =

                Type:    'QAM Modulator'
                   M:    8
         PhaseOffset:    0
       Constellation:    [1x8 double]
         SymbolOrder:    'Binary'
       SymbolMapping:    [0 1 2 3 4 5 6 7]
           InputType:    'Integer'


g =

                Type:    'QAM Demodulator'
                   M:    8
         PhaseOffset:    0
       Constellation:    [1x8 double]
         SymbolOrder:    'Binary'
       SymbolMapping:    [0 1 2 3 4 5 6 7]
          OutputType:    'Integer'
        DecisionType:    'Hard decision'
```

1.3.7. Вывод на дисплей изображений созвездий сигналов

Чтобы вывести на экран монитора созвездие сигнала, связанное с процессом модуляции, следует выполнить следующие процедуры.

1. Если размер алфавита для процесса модуляции составляет M, тогда создается сигнал [0:M – 1]. Этот сигнал представляет все возможные входы на модулятор.

2. Используются подходящие функции к процессу модуляции функции, чтобы осуществить модуляцию сигнала. Если нужно, можно изменить масштаб представления созвездия. Результат выполнения этой процедуры – получение всех возможных точек созвездия сигнала.

3. Применить функцию scatterplot, чтобы создать модулированный выход для построения диаграммы.

Пример 4. Построить созвездия сигналов 16PSK, 32QAM, созвездия сигнала, представленного кодом Грея, созвездия с обычной QAM.

Решение:

Построим созвездие для сигнала с 16PSK, имеющего 16 точек в соответствии с кодом, приведенным ниже. Диаграмма представлена на рис. 2.

```
>>M = 16;
>>x = [0:M-1];
>>scatterplot(modulate(modem.pskmod(M),x));
```

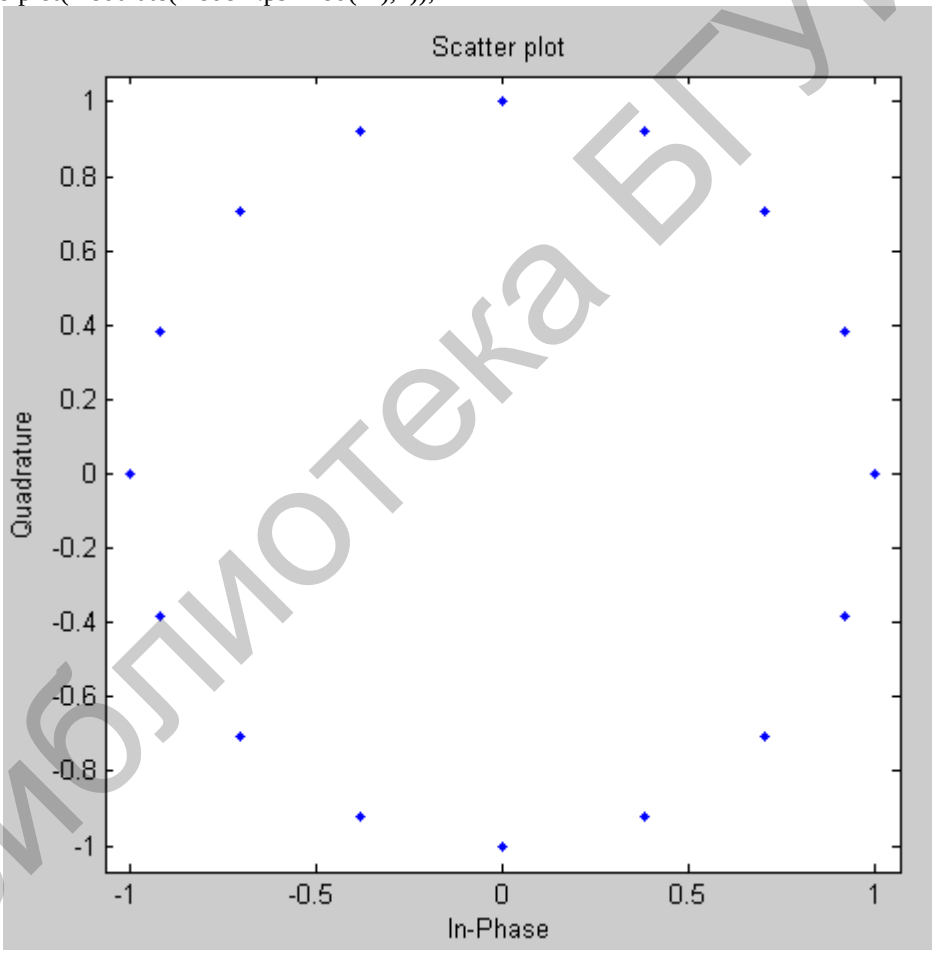


Рис. 2. Созвездие для сигнала с фазовой манипуляцией 16PSK

Построим созвездие для сигнала с квадратурной амплитудной модуляцией 32QAM. Это созвездие имеет 32 точки и пиковую мощность 1 Вт. Проиллюстрируем также механизм обозначения чисел при формировании входа модулятора. Составим программу для выполнения этих операций. Имеем

```
>>M = 32;
>>x = [0:M-1];
```

```
>>y = modulate(modem.qammod(M),x);
>>scale = modnorm(y,'peakpow',1);
>>y = scale*y; % установим масштаб созвездия.
>>scatterplot(y); % Диаграмма масштабированного созвездия

% Включим текстовый комментарий для обозначения точек созвездия числами.
>>hold on; % Make sure the annotations go in the same figure.
>>for jj=1:length(y)
>>text(real(y(jj)),imag(y(jj)),['  ' num2str (jj-1)]);
>>end
>>hold off;
```

Полученное созвездие приведено на рис. 3.

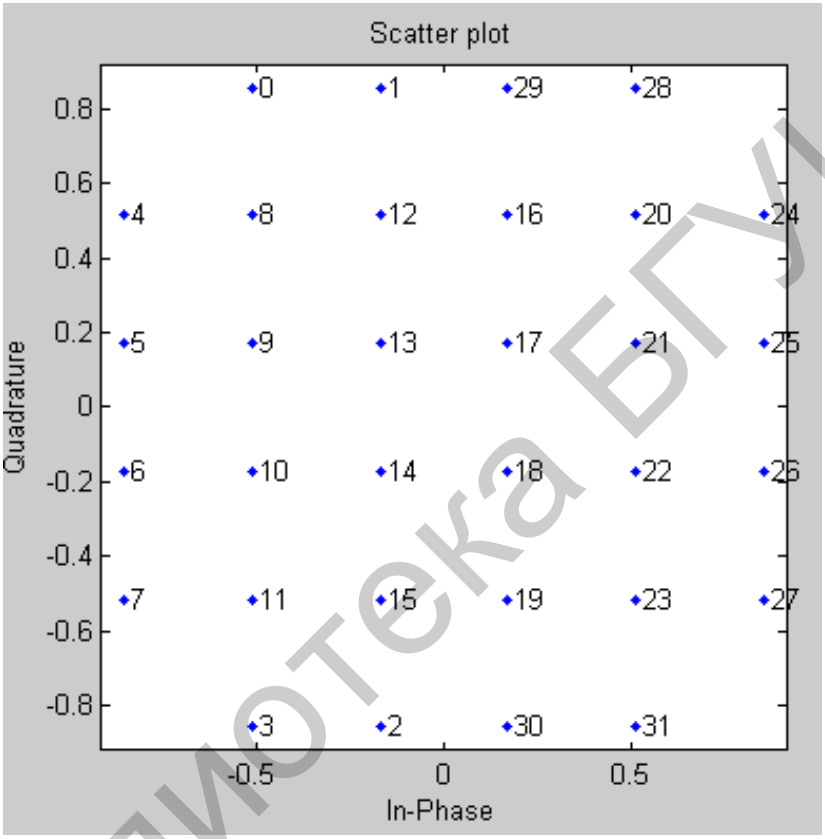


Рис. 3. Созвездие для сигнала с квадратурной амплитудной манипуляцией 32QAM с обозначением точек созвездия числами

Построим созвездие для сигнала 8QAM, представленное кодом Грея. Диаграмма, представленная на рис. 4, описывает созвездие с указанием наименований точек созвездия числами. Созвездие использует код Грея.

```
>>M = 8;
>>x = [0:M-1];
>>y = modulate(modem.qammod('M',M,'SymbolOrder','Gray'),x);
%Построим изображение созвездия сигнала, кодированного кодом Грея.
>>scatterplot(y,1,0,'b.'); % Dots for points.
>>% Включим выполнение обозначений точек созвездия числами.
>>hold on; % Make sure the annotations go in the same figure.
>>annot = dec2bin([0:length(y)-1],log2(M));
>>text(real(y)+0.15,imag(y),annot);
>>axis([-4 4 -4 4]);
>>title('Constellation for Gray-Coded 8-QAM');
>>hold off;
```

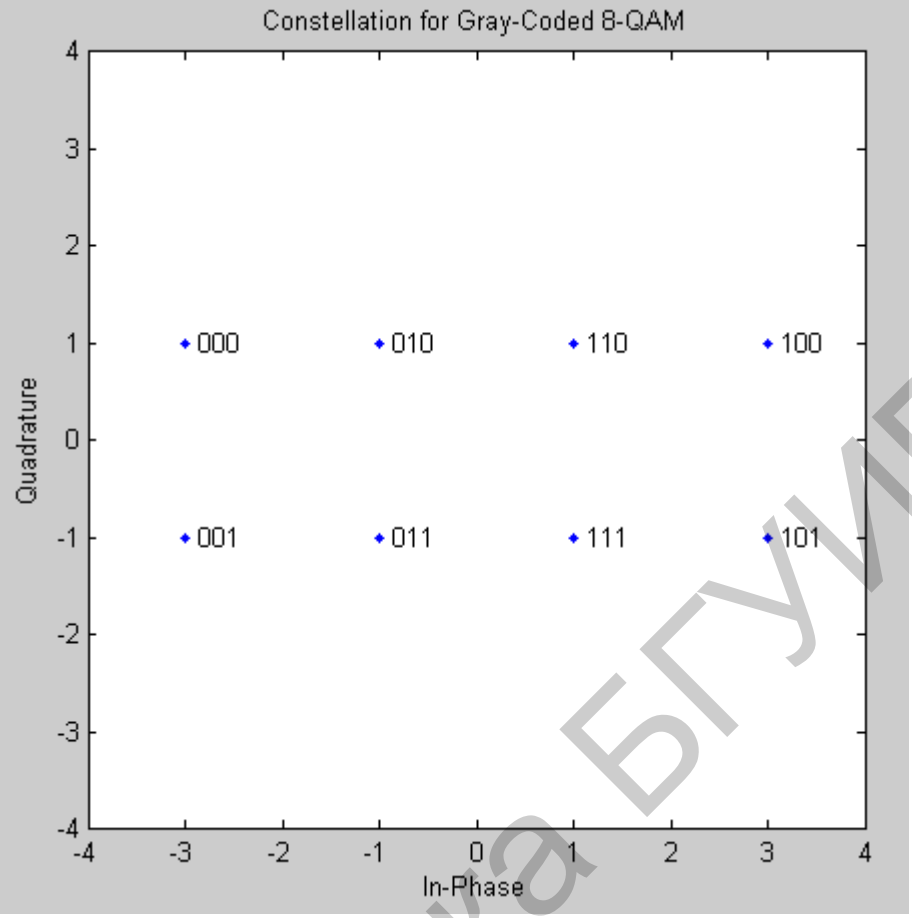


Рис. 4. Созвездие для сигнала 8QAM с квадратурной амплитудной манипуляцией и кодированием Грея с обозначением точек созвездия числами

Построим созвездие для обычной квадратурной амплитудной модуляции. Приведенный ниже код описывает также структуру созвездия, показанного на рис. 5.

```
% Опишем структуру созвездия.
>>inphase = [1/2 -1/2 1 0 3/2 -3/2 1 -1];
>>quadr = [1 1 0 2 1 1 2 2];
>>inphase = [inphase; -inphase]; inphase = inphase(:);
>>quadr = [quadr; -quadr]; quadr = quadr(:);
>>const = inphase + j*quadr;

% Строим созвездие
>>scatterplot(const,1,0,'*');
>>hold on;
>>axis([-3 3 -3 3]);
>>title('Customized Constellation for QAM');
>>hold off;
```

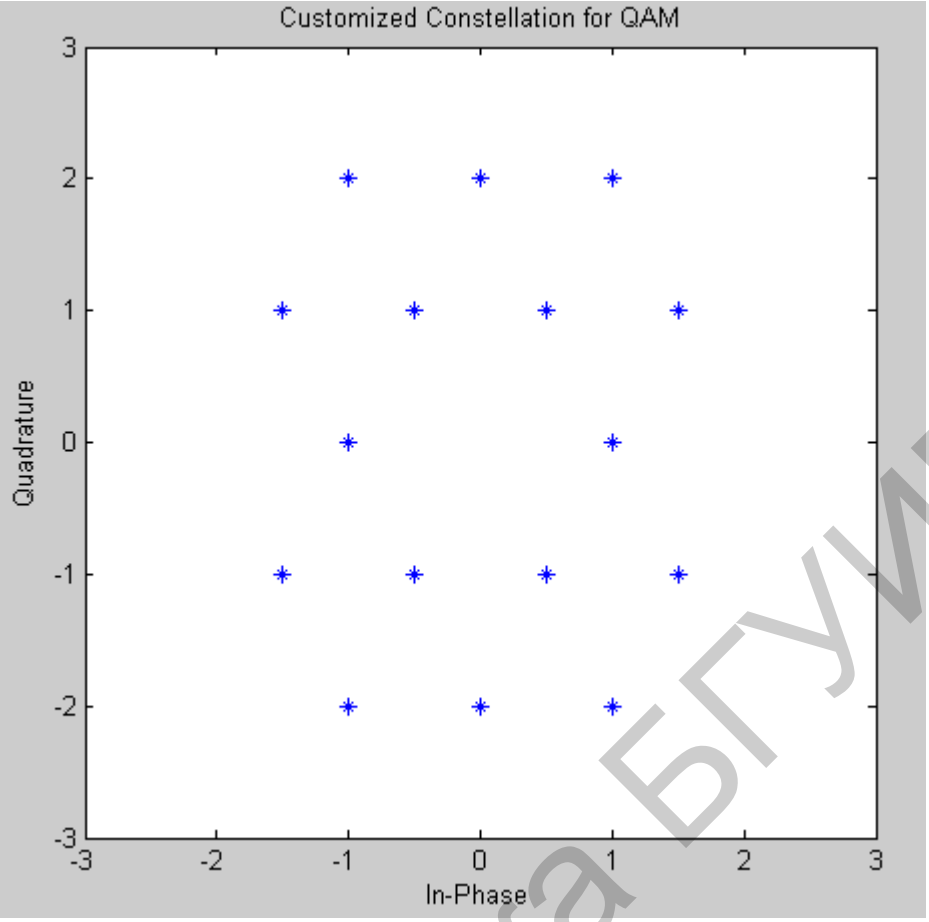


Рис. 5. Созвездие для сигнала 16QAM
с квадратурной амплитудной манипуляцией

1.3.8. Использование функции modnorm при построении созвездий сигналов с цифровой манипуляцией

Функция modnorm представляет собой инструмент для нормализации выходного сигнала модуляции. Синтаксис функции modnorn имеет вид

>>scale = modnorm(const, 'avpow', avpow)

>>scale = modnorm(const, 'peakpow', peakpow)

Описание функции modnorm

Функция scale = modnorm(const, 'avpow', avpow) возвращает returns коэффициент масштабирования выходного сигнала модулятора сигналов, нормализующий его выход для сигналов с PAM и QAM, таким образом, что средняя мощность сигнала представляется параметром avpow (watts). Параметр const – это вектор, характеризующий созвездие, используемое для создания коэффициента масштабирования. Функция предполагает, что сигнал, который должен быть нормирован, имеет минимальное расстояние, равное 2.

Функция scale = modnorm(const, 'peakpow', peakpow) возвращает фактор масштабирования для нормирования PAM или QAM выхода модулятора таким образом, что его пиковая мощность соответствует параметру peakpow (watts). Покажем на примере, как использовать эту функцию.

Пример 5. Использовать функцию modnorm для передачи квадратурно модулированного сигнала, имеющего пиковую мощность 1 Вт. Записать коды MATLAB для выполнения программы.

Решение:

```
>>M = 16; % Alphabet size (размер алфавита)
>>const = qammod([0:M-1],M); % Генерируем созвездие.
>>x = randint(1,100,M);
>>scale = modnorm(const,'peakpow',1); % Рассчитываем масштабирующий коэффициент.
>>y = scale * qammod(x,M); % Модулируем и масштабируем
>>ynoisy = awgn(y,10); % передаем сигнал по каналу с шумом.
>>ynoisy_unscaled = ynoisy/scale; % масштаб на приемном конце канала связи.
>>z = qamdemod(ynoisy_unscaled,M); % выполняем демодуляцию
% Посмотрим, как проявляется масштабирование для представления диаграммы созвездия
>>h = scatterplot(const,1,0,'ro'); % Unscaled constellation
>>hold on; % Next plot will be in same figure window.
>>scatterplot(const*scale,1,0,'bx',h); % Scaled constellation
>>hold off;
```

На диаграмме, приведенной ниже на рис. 6, изображенные на диаграмме символы и отмеченные кружочками соответствуют исходному созвездию сигнала с QAM, в то время как символы, отмеченные крестиками, соответствуют созвездию сигнала, которое масштабировано функцией modnorm. Канал в этом примере переносит точки масштабированного созвездия.

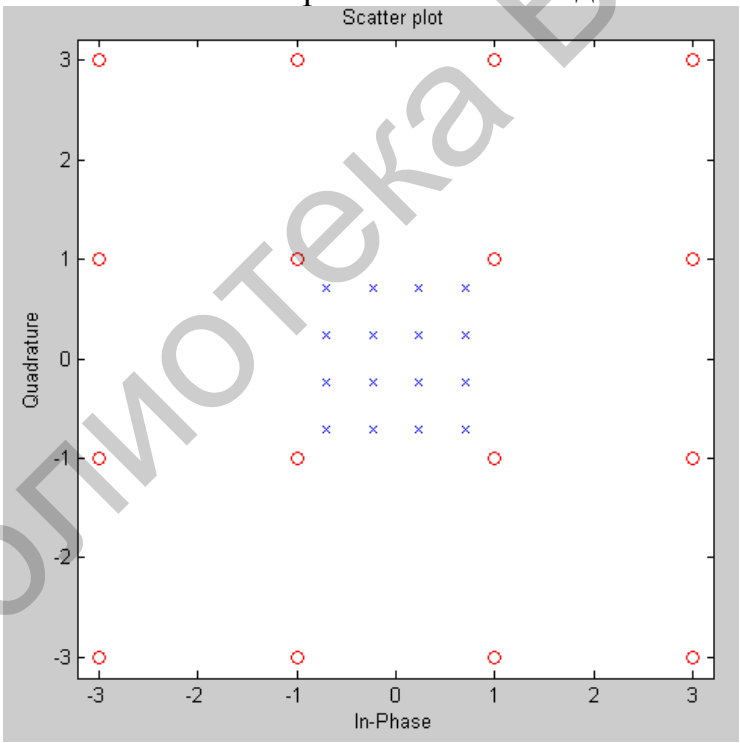


Рис. 6. Изображения исходного и масштабированного созвездий

2. Подготовка к выполнению лабораторной работы

1. Изучить краткие теоретические сведения, приведенные в описании этой лабораторной работы.

2. Изучить по конспекту лекций структуру сложных сигналов с фазовой и частотной манипуляцией.

3. Составить программы для формирования созвездий, изучаемых в данной работе сигналов, а также нарисовать созвездия сигналов, которые вы должны изучить в процессе выполнения данной работы.

3. Выполнение лабораторной работы

Задание 1. Выполните эксперименты, описанные в примере 1 предыдущего пуннкта данной работы, при различных значениях частоты исходного и несущего сигналов. Получите и зарисуйте результаты проведенных измерений.

Задание 2. Выполните эксперименты, описанные в примерах 2 – 5 предыдущего пункта настоящей лабораторной работы, затем перейдите к выполнению данного пункта. Повторите его для разных значений частоты модулирующего сигнала, составив программу в виде функции.

3.1. Минимальная частотная модуляция

Минимальная частотная манипуляция в пакете MATLAB моделируется с использованием функции mskmod.

Синтаксис этой функции имеет четыре варианта и их можно представить в виде

y = mskmod(x,nsamp)

y = mskmod(x,nsamp,dataenc)

y = mskmod(x,nsamp,dataenc,ini_phase)

[y,phaseout] = mskmod(...)

Функция y = mskmod(x,nsamp) выводит значения комплексной огибающей y, полученной при модуляции его сообщением x, используя дифференциально кодированную частотную модуляцию с минимальным частотным сдвигом. Элементы x должны быть либо нулем, либо единицей. Параметр nsamp обозначает число отсчетов, приходящихся на символ в модулированном сигнале y и должен имеет целое положительное значение. Начальная фаза сигнала с минимальным частотным сдвигом должна быть равна нулю. Если x представляет собой матрицу со многими строками и столбцами, функция y = mskmod(x,nsamp) трактует столбцы как независимые каналы и обрабатывает их независимо друг от друга.

Функция y = mskmod(x,nsamp,dataenc) характеризует метод кодирования данных для MSK. Параметр dataenc может быть либо 'diff' дифференциально кодированной MSK, либо 'nondiff' недифференциально кодированной MSK.

Функция y = mskmod(x,nsamp,dataenc,ini_phase) характеризует начальную фазу сигнала в модуляторе MSK . Параметр ini_phase – это вектор-строка, длина которой соответствует числу каналов в сигнале y и значения которых равны целому числу, умноженному на π/2. Чтобы избежать переполнения по умолчанию значение параметра dataenc устанавливается равным [y, phaseout].

Функция [y,phaseout] = mskmod(...) возвращает конечную фазу модулированного сигнала y. Это полезно для поддержания фазы непрерывной, когда осуществляется модуляция потока битов с дифференциально кодированной MSK. Параметр phaseout имеет ту же размерность, что и начальная фаза на входе и допускает значения 0, π/2, π, и 3*π/2.

Приведем код, с помощью которого создается глазковая диаграмма сигнала с минимальным частотным сдвигом. Результат работы приведенного кода показан на рис. 7.

```
>>x = randint(99,1); % Random signal
>>y = mskmod(x,8,[],pi/2);
>>y = awgn(y,30,'measured');
>>eyediagram(y,16);
```

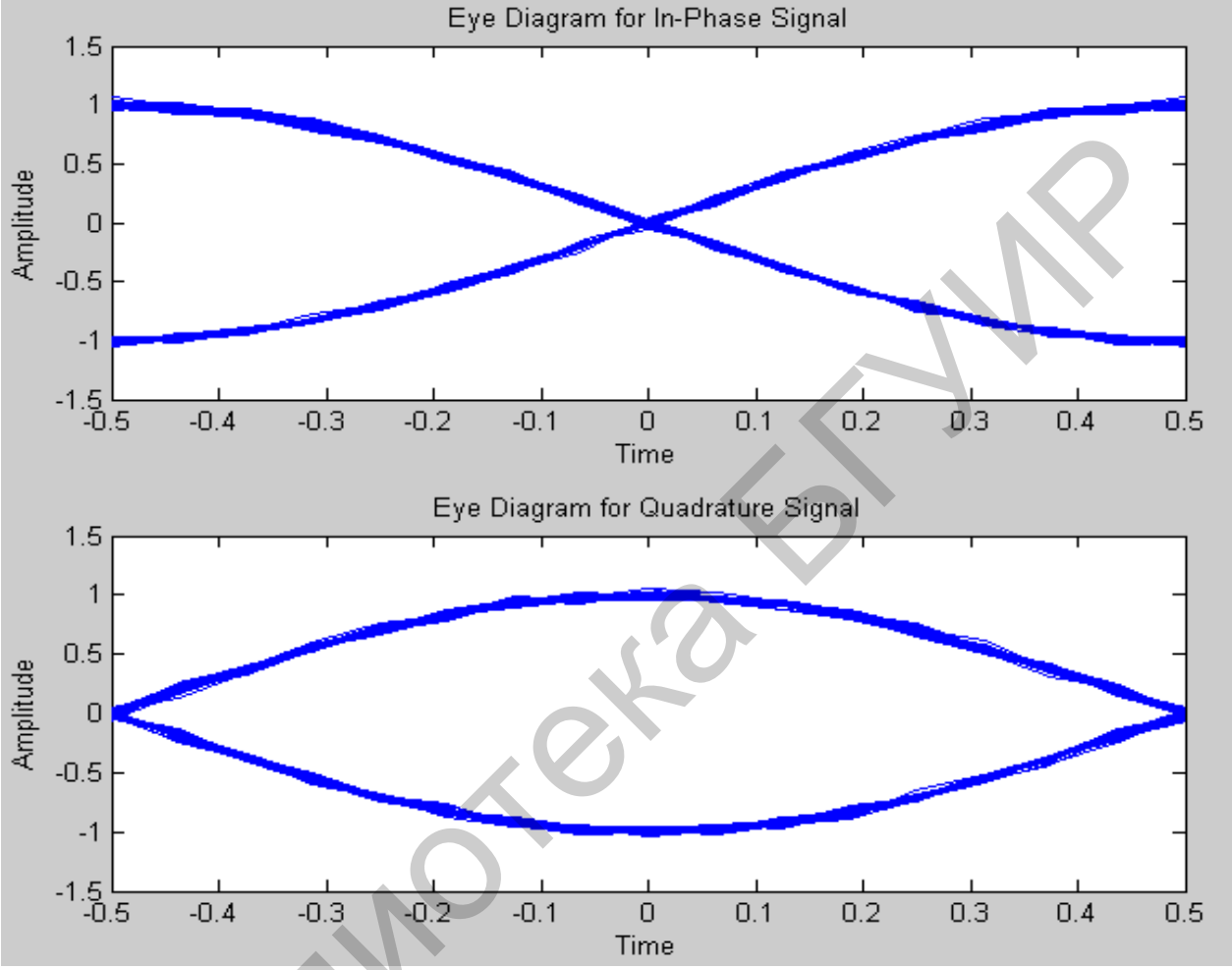


Рис. 7. Глазковые диаграммы синфазной и квадратурной составляющих сигнала

Задание 3. Выполните эксперимент по построению глазковой диаграммы 2 – 3 раза при разных значениях частоты модулирующего сигнала в соответствии с методикой, изложенной в приведенном здесь примере.

3.2. Демодуляция сигнала с минимальной частной манипуляцией

Минимальная частотная манипуляция выполняется с помощью функции mskmod, а демодуляция полученного с помощью этой функции сигнала выполняется с помощью функции mskmod.

Синтаксис функции имеет вид

```
>>z = mskdemod(y, nsamp)
>>z = mskdemod(y, nsamp,  dataenc)
>>z = mskdemod(y, nsamp, dataenc, ini_phase)
>>z = mskdemod(y, nsamp, dataenc,  ini_phase, ini_state)
```

```
>>[z, phaseout] = mskdemod(...)
>>[z, phaseout,stateout] = mskdemod(...)
```

Приведем краткое описание функции демодуляции сигнала с минимальной частотной манипуляцией.

Функция z = mskdemod(y,nsamp) демодулирует комплексную огибающую сигнала, использующую метод дифференциально кодированной частотной модуляции с минимальным сдвигом. Параметр nsamp обозначает число отсчетов, приходящихся на один символ, и должен иметь целое положительное значение. Начальная фаза модулятора равна нулю. Если y представляет собой матрицу со многими строками и столбцами, функция z = mskdemod(x,nsamp) трактует столбцы как независимые каналы и обрабатывает их независимо друг от друга.

Функция z = mskdemod(y,nsamp,dataenc) характеризует метод кодирования данных для MSK. Параметр dataenc может быть либо 'diff' для дифференциально кодированной MSK либо 'nondiff' для недифференциально кодированной MSK.

Функция z = mskdemod(y,nsamp,dataenc,ini_phase) характеризует начальную фазу демодулятора. Параметр ini_phase – это вектор-строка, длина которого равна числу каналов в частотно-модулированном сигнале и значения которого умножаются на π/2. Во избежание переполнения значение dataenc по умолчанию устанавливается равным [].

Функция z = mskdemod(y,nsamp,dataenc,ini_phase,ini_state) характеризует начальную фазу демодулятора. Параметр ini_state содержит последнюю половину символа предварительно принятого сигнала. Параметр ini_state – это nsamp-by-C матрица, где C представляет число каналов в частотно-модулированном сигнале *y*.

Функция [z,phaseout] = mskdemod(...) возвращает конечную фазу сигнала y, которая имеет важное значение для демодуляции сигналов в будущем. Фаза на выходе имеет ту же размерность, что и ini_phase на входе, и допускает значения 0, π/2, π, and 3*π/2.

[z,phaseout,stateout] = mskdemod(...) возвращает конечное значение nsamp сигнала *y*, которое будет полезным при демодуляции первых символов в сигналах будущего. Параметр stateout имеет ту же размерность, что и ini_state на входе.

Пример 6. Модулировать и демодулировать сигналы с помощью функции mskmod и mskdemod, используемых в единой системе. (Чтобы обеспечить непрерывность перехода от одной итерации к другой, синтаксис mskmod и mskdemod использует начальные фазы и/или состояния аргументов на входе и выходе).

Решение:

```
% Определим параметры.
>>numbits = 99; % Number of bits per iteration
>>numchans = 2; % Number of channels (columns) in signal
>>nsamp = 16; % Number of samples per symbol
% Инициализируем процесс.
>>numerrs = 0; % Number of bit errors seen so far
>>demod_ini_phase = zeros(1,numchans);  % Modulator phase
>>mod_ini_phase = zeros(1,numchans);    % Demodulator phase
>>ini_state = complex(zeros(nsamp,numchans)); % Demod. state
```

```
>>% Главная петля
>>for iRuns = 1 : 10
>>x = randint(numbits,numchans); % Binary signal
>>[y,phaseout] = mskmod(x,nsamp,[],mod_ini_phase);
>>mod_ini_phase = phaseout; % For next mskmod command
>>[z, phaseout, stateout] = ...
>>mskdemod(y,nsamp,[],demod_ini_phase,ini_state);
>>ini_state = stateout;  % For next mskdemod command
>>demod_ini_phase = phaseout; % For next mskdemod command
>>numerrs = numerrs + biterr(x,z); % Cumulative bit errors
>>end
>>disp(['Total number of bit errors = ' num2str(numerrs)])
```

Сформируем результаты на выходе. Получаем общее число битовых ошибок, равное нулю.

Total number of bit errors = 0

Задание 4. Выполните эксперименты, описанные в примере 6 данной работы, 2 – 3 раза при разных значениях частоты модулирующего и несущего сигналов.

3.3. Дифференциальная фазовая манипуляция сигнала с помощью функции dpskmod

Синтаксис этой функции dpskmod, с помощью которой выполняется моделирование сигналов с дифференциальной фазовой манипуляцией, имеет три варианта, которые представлены следующим образом.

```
>>y = dpskmod(x,M)
>>y = dpskmod(x,M,phaserot)
>>y = dpskmod(x,M,phaserot,symbol_order)
```

Функция y = dpskmod(x,M) выводит комплексную огибающую сигнала y сигналом сообщения x и использует дифференциальную фазовую модуляцию. Параметр M – размер алфавита, который должен быть целым числом. Сигнал сообщения должен состоять из целых чисел, находящихся в промежутке от 0 до M – 1. Если сигнал x представлен матрицей с несколькими строками и столбцами, тогда функция обрабатывает столбцы независимо.

Функция y = dpskmod(x,M,phaserot) характеризует поворот фазы при модуляции, который измеряется в радианах. В случае когда общий фазовый сдвиг, приходящийся на один символ, равен сумме phaserot и фаза генерируется дифференциальной модуляцией.

Функция y = dpskmod(x,M,phaserot,symbol_order) характеризует, как функция присоединяет бинарные слова к соответствующим числам. Если параметр symbol_order представляет собой множество 'bin' (по умолчанию), функция использует естественный порядок бинарного кодирования. Если параметр symbol_order принадлежит множеству 'gray', тогда он использует кодирование Грея,

Приведем пример, показывающий, как формируется выход функции dpskmod.

Пример 7. На примере созвездия сигнала с DPSK показать возможные пути передачи на диаграмме, на которой видно, как передается информация от одного символу к другому при выполнении модуляции.

Решение:

Приведем соответствующий код MATLAB, по которому получен результат, показанный на рис. 8. Итак, имеем

```
>>M = 4; % Используем  DQPSK в этом примере, так что  M = 4.
>>x = randint(500,1,M,13); % Формируем случайные данные
>>y = dpskmod(x,M,pi/8); % Модулируем, используя ненулевую начальную фазу.
>>plot(y) % Представим все точки на диаграмме, используя линии для их соединения.
```

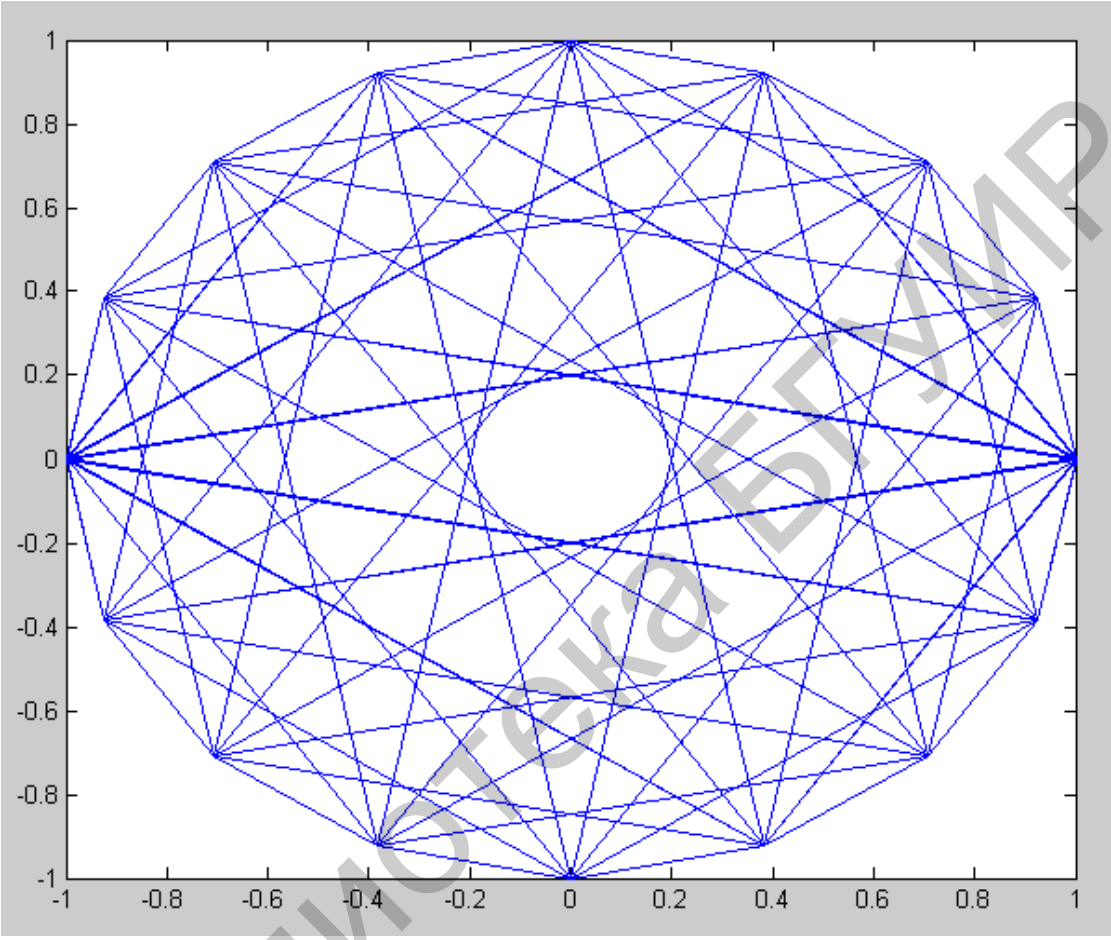


Рис. 8. Диаграмма возможных путей перехода между точками созвездия при дифференциальной фазовой модуляции

3.4. Деманипуляция дифференциальной фазовой манипуляции с помощью функции dpskdemod

Синтаксис функции dpskdemod имеет три варианта:

```
>>z = dpskdemod(y,M)
>>z = dpskdemod(y,M,phaserot)
>>z = dpskdemod(y,M,phaserot,symbol_order).
```

Функция z = dpskdemod(y,M) демодулирует комплексную огибающую сигнала с дифференциальной фазовой манипуляцией. Параметр M – это размер алфавита, M должен быть целым числом. Если y – матрица с несколькими строками и столбцами, функция dpskdemod обрабатывает столбцы независимо друг от друга.

Заметим, что первый элемент выходного сигнала z или первая строка этого сигнала z, если z – матрица с несколькими строками и столбцами, представляют начальные условия, потому что дифференциальный алгоритм сравнивает два последовательных элемента модулированного сигнала.

Функция z = dpskdemod(y,M,phaserot) характеризует поворот фазы при модуляции, который измеряется в радианах. В случае когда общий фазовый сдвиг, приходящийся на один символ, равен сумме phaserot и фаза генерируется дифференциальной манипуляцией.

Функция z = dpskdemod(y,M,phaserot,symbol_order) характеризует, как функция присоединяет бинарные слова к соответствующим числам. Если параметр symbol_order представляет собой множество 'bin' (по умолчанию), функция использует естественный порядок бинарного кодирования. Если параметр symbol_order принадлежит множеству 'gray', тогда он использует кодирование Грея.

Приведем пример, чтобы проиллюстрировать работу этой функции.

Пример 8. Проиллюстрировать факт, что первые выходные символы демодулятора дифференциальной фазовой модуляции – это начальные условия, которые предшествуют полезной информации.

Решение:

```
>>M = 4; % Размер алфавита
>>x = randint(1000,1,M); % Случайное сообщение
>>y = dpskmod(x,M); % Модуляция
>>z = dpskdemod(y,M); % Демодуляция
% Проверим, правильно ли демодулятор восстановил исходное сообщение
>>s1 = symerr(x,z) %  Ожидается одна символьная ошибка, т.е. первый символ.
>>s2 = symerr(x(2:end),z(2:end)) % Игнорируем первый символ, ожидаем отсутствия ошибок.
```

Выход приведенная программа выдает привычным для системы MATLAB образом:

```
s1 =
    1
s2 =
    0
```

Задание 5. Выполните эксперименты, описанные в примерах 7 и 8 данной работы, а затем промодулируйте и демодулируйте сигнала с дифференциальной фазовой манипуляцией 2 – 3 раза при разных значениях частоты модулирующего и несущего сигналов.

3.5. Фазовая манипуляция, реализуемая функцией pskmod

Синтаксис функции pskmod, с помощью которой можно моделировать фазовую манипуляцию, имеет три варианта:

>>y = pskmod(x, M)

>>y = pskmod(x, M, ini_phase)

>>y = pskmod(x, M, ini_phase,symbol_order).

Функция y = pskmod(x,M) выводит комплексную огибающую модулированного по фазе сообщением x сигнала y, используя фазовую манипуляцию. Число M – размер алфавита и должен выражаться целой степенью двойки. Сиг-

нал сообщения должен состоять из целых чисел, расположенных между 0 и M –1. Начальная фаза сигнала при модуляции равна нулю. Если x представляет собой матрицу с несколькими строками и колонками, функция pskmod обрабатывает столбцы независимо.

Функция y = pskmod(x, M, ini_phase) характеризует начальную фазу модуляции в радианах.

Функция y = pskmod(x,M,ini_phase,symbol_order) показывает, как функция присоединяет бинарные слова к соответствующим числам. Если параметр symbol_order – принадлежит множеству 'bin' (по умолчанию), функция использует естественный порядок бинарного кодирования. Если параметр symbol_order – принадлежит множеству 'gray', тогда используется созвездие с кодированием Грея.

3.6. Фазовая деманипуляция, реализуемая функцией pskdemod

Синтаксис функции pskdemod имеет три варианта:

```
>>z = pskdemod(y, M)
>>z = pskdemod(y, M, ini_phase)
>>z = pskdemod(y,  M, ini_phase,symbol_order).
```

Функция z = pskdemod(y,M) демодулирует комплексную огибающую y модулированного сигнала с фазовой манипуляцией. Число M – размер алфавита и должен выражаться целой степенью двойки. Начальная фаза сигнала при модуляции равна нулю. Если x представляет собой матрицу с несколькими строками и колонками, функция pskmod обрабатывает столбцы независимо.

Функция z = pskdemod(y,M,ini_phase) характеризует начальную фазу модуляции в радианах.

Функция z = pskdemod(y,M,ini_phase,symbol_order) показывает, как функция присоединяет бинарные слова к соответствующим числам. Если параметр symbol_order – принадлежит множеству 'bin' (по умолчанию), функция использует естественный порядок бинарного кодирования. Если параметр symbol_order – принадлежит множеству 'gray', тогда используется созвездие с кодированием Грея.

Пример 9. Провести сравнение фазовой манипуляции и фазовой амплитудной модуляции (phase amplitude modulation, PAM), чтобы увидеть, что PSK более чувствительна к фазовому шуму.

Решение:

Этот результат ожидается интуитивно, потому что созвездие фазовой модуляции представляет собой окружность, а созвездие PAM является линейным.

```
>>len = 10000; % Number of symbols
>>M = 16; % Size of alphabet
>>msg = randint(len,1,M); % Original signal
>>% Осуществляем модуляцию, используя и PSK, и PAM,
>>%чтобы сравнить два метода.
>>txpsk = pskmod(msg,M);
>>txpam = pammod(msg,M);
>>% Возмущение фазы модулированного сигналов.
>>phasenoise = randn(len,1)*.015;
```

```
>>rxpsk = txpsk.*exp(j*2*pi*phasenoise);
>>rxpam = txpam.*exp(j*2*pi*phasenoise);
>>% Построение диаграмм принятых сигналов.
>>scatterplot(rxpsk); title('Noisy PSK Scatter Plot')
>>scatterplot(rxpam); title('Noisy PAM Scatter Plot')
>>% Демодулирование принятых сигналов.
>>recovpsk = pskdemod(rxpsk,M);
>>recovpam = pamdemod(rxpam,M);
>>% Расчет числа символьных ошибок в каждом случае.
>>numerrs_psk = symerr(msg,recovpsk)
>>numerrs_pam = symerr(msg,recovpam)
```

Диаграммы рассеяния, полученные для сигналов с PSK и PAM, приведены на рис. 9 и 10 соответственно.

Заметим, что результаты, полученные при выполнении работы, могут отличаться от полученных здесь, так сигнал формируется в каждом случае случайным образом.

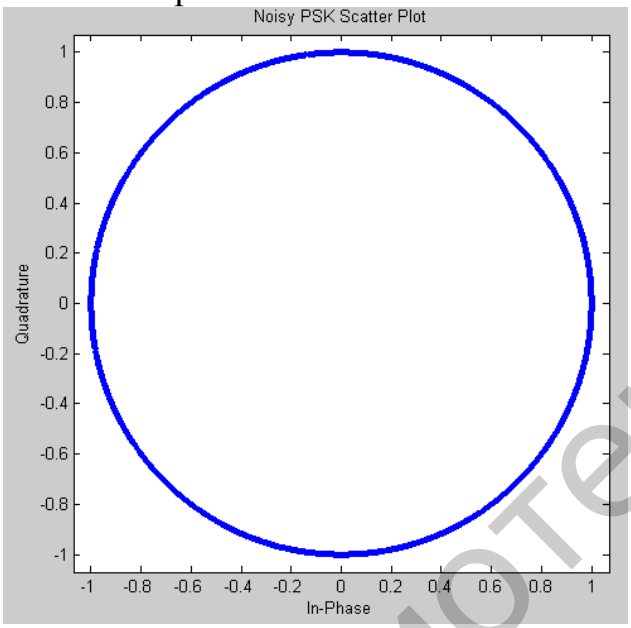


Рис. 9. Диаграмма рассеяния, полученная для сигнала с фазовой манипуляцией

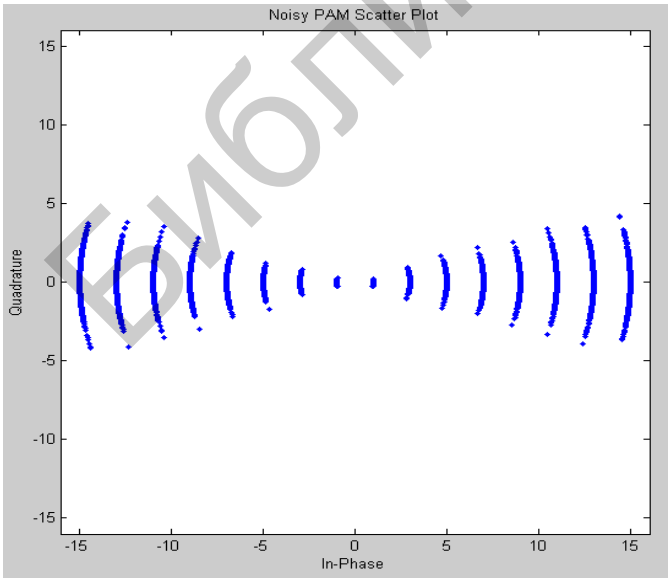


Рис. 10 . Диаграмма рассеяния, полученная для сигнала с амплитудной манипуляцией

```
numerrs_psk =
  374
numerrs_pam =
    1
```

Задание 6. Выполните эксперименты, описанные в примерах 9, 10 и 11 данной работы, а затем промодулируйте и демодулируйте сигналы с дифференциальной фазовой манипуляцией 2 – 3 раза при разных значениях частоты модулирующего и несущего сигналов.

3.7. Частотная манипуляция, реализуемая функцией fskmod

Синтаксис функции fskmod, использующейся в системе MATLAB для моделирования сигнала с частотной манипуляцией, имеет 4 варианта использования следующего вида:

```
>>y = fskmod(x, M, freq_sep, nsamp)
>>y = fskmod(x, M, freq_sep, nsamp,Fs)
>>y = fskmod(x, M, freq_sep,nsamp, Fs,phase_cont)
>>y = FSKMOD(x, M, freq_sep, nsamp, Fs,phase_cont, symbol_order)
```

Функция y = fskmod(x,M,freq_sep,nsamp) выводит комплексную огибающую модулированного по частоте сообщением *x* сигнала *y*, используя частотную манипуляцию. Число M – размер алфавита и должен выражаться целой степенью двойки. Сигнал сообщения должен состоять из целых чисел, расположенных между 0 и M – 1. Параметр freq_sep устанавливает разделение между последовательными частотами и измеряется в герцах. Параметр nsamp обозначает число отсчетов, приходящихся на один символ в модулированном сигнале, и должен быть целым положительным числом, большим чем 1. Скорость дискретизации измеряется в герцах. По теореме Найквиста параметр freq_sep и M должны удовлетворять условию (M – 1)*freq_sep <= 1. Если *x* представляет собой матрицу с несколькими строками и колонками, функция pskmod обрабатывает столбцы независимо.

Функция y = fskmod(x,M,freq_sep,nsamp,Fs) характеризует скорость дискретизации модулированного сигнала y в герцах. Вследствие этого теорема Найквиста требует, чтобы максимальная частота не должна быть более чем Fs/2, на входе системы должно выполняться условие (M – 1)*freq_sep <= Fs.

Функция y = fskmod(x,M,freq_sep,nsamp,Fs,phase_cont) соответствует фазовой непрерывности. Установить параметр phase_cont можно в значение 'cont', чтобы усилить фазовую непрерывность на границах символов в модулированном сигнале *y* или в значение 'discont', чтобы избежать форсирования фазовой непрерывности. По умолчанию устанавливается значение параметра phase_cont в значение 'cont'.

Функция y = FSKMOD(x,M,freq_sep,nsamp,Fs,phase_cont,symbol_order) показывает, как функция присоединяет бинарные слова к соответствующим целым числам. Если параметр symbol_order is установлен в 'bin' (по умолчанию), функция использует естественный порядок бинарного кодирования. Если па-

раметр symbol_order принадлежит множеству 'gray', тогда используется созвездие с кодированием Грея.

Пример 10. Проиллюстрировать синтаксис функции fskmod, используя случайный сигнал, и сформировать спектр модулированного по частоте сигнала.

Решение:

Продемонстрируем это с помощью кодов MATLAB и результаты выполнения этих команд покажем на рис. 11. Имеем

```
>>M = 4;
>>freqsep = 8;
>>nsamp = 8;
>>Fs = 32;
>>x = randint(1000,1,M); % Образуем случайный сигнал.
>>y = fskmod(x,M,freqsep,nsamp,Fs); % Осуществим частотную манипуляцию
>>ly = length(y);
% Создадим диаграмму с преобразованием Фурье.
>>freq = [-Fs/2 : Fs/ly : Fs/2 – Fs/ly];
>>Syy = 10*log10(fftshift(abs(fft(y))));
>>plot(freq,Syy)
```

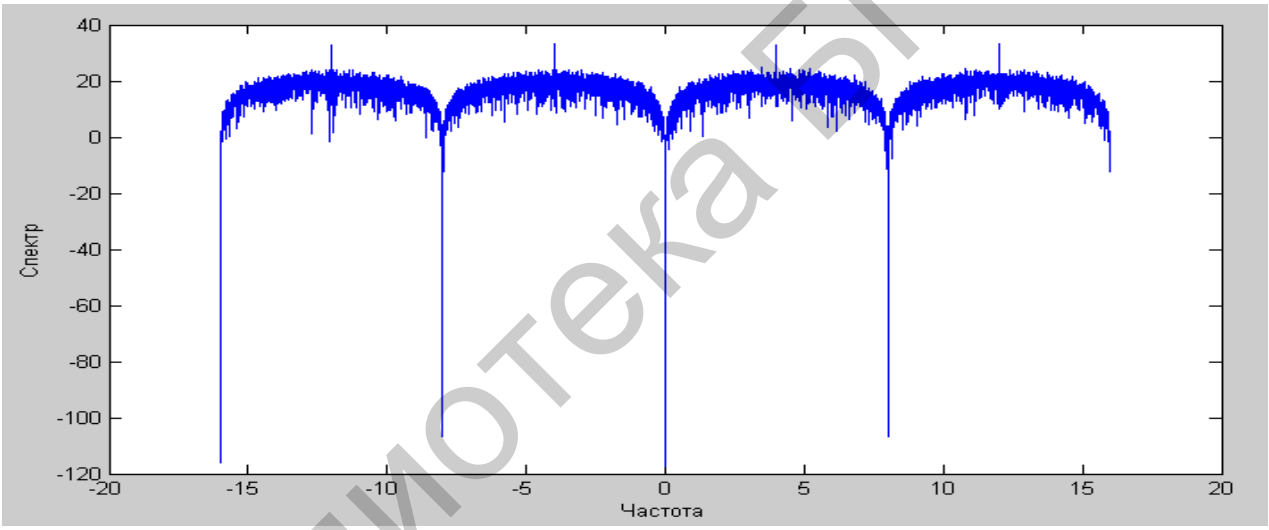


Рис. 11. Спектр сигнала, модулированного случайным процессом по частоте

3.8. Частотная деманипуляция, реализуемая функцией fskdemod

Синтаксис функции fskdemod имеет 3 варианта использования:

```
>>z = fskdemod(y,M,freq_sep,nsamp)
>>z = fskdemod(y,M,freq_sep,nsamp,Fs)
>>z = fskdemod(y,M,freq_sep,nsamp,Fs,symbol_order)
```

Функция z = fskdemod(y,M,freq_sep,nsamp) некогерентно демодулирует комплексную огибающую модулированного сигнала y, использующую метод частотной манипуляции. Число M – размер алфавита и должен выражаться целой степенью двойки. Несущая частота сигнала выбирается равной 1 Гц. Параметр аргумента freq_sep представляет собой число отсчетов, приходящихся на один символ и должен быть целым положительным числом, большим чем 1.

Если *y* представляет собой матрицу с несколькими строками и колонками, функция pskdemod обрабатывает столбцы независимо.

Функция z = fskdemod(y,M,freq_sep,nsamp,Fs) характеризует частоту дискретизации, которая измеряется в герцах.

Функция z = fskdemod(y,M,freq_sep,nsamp,Fs,symbol_order) показывает, как функция присоединяет бинарные слова к соответствующим числам. Если параметр symbol_order is установлен в 'bin' (по умолчанию), функция использует естественный порядок бинарного кодирования. Если параметр symbol_order принадлежит множеству 'gray', тогда используется созвездие с кодированием Грея.

Пример 11. Проиллюстрировать частотную манипуляцию и деманипуляцию, которые выполняются над сигналом, распространяющимся в условиях помех, заданных аддитивным белым гауссовским шумом.

Решение:

Выполним следующие коды MATLAB.

```
>>M = 2;
>>k = log2(M);
>>EbNo = 5;
>>Fs = 16;
>>nsamp = 17;
>>freqsep = 8;
>>msg = randint(5000,1,M); % случайный сигнал
>>txsig = fskmod(msg,M,freqsep,nsamp,Fs); % Осуществляем модуляцию.
>>msg_rx  = awgn(txsig,EbNo+10*log10(k)-10*log10(nsamp),...
 'measured',[],'dB'); % Канал с аддитивным белым гауссовским шумом
>>msg_rrx = fskdemod(msg_rx,M,freqsep,nsamp,Fs);  % Осуществляем деманипуляцию
>>[num,BER] = biterr(msg,msg_rrx) % скорость битовой ошибки
>>BER_theory = berawgn(EbNo,'fsk',M,'noncoherent') % Теоретическое значение скорости битовой ошибки.
```

Результат работы этой программы показан ниже. Однако результаты, полученные при повторном выполнении этой программы, могут отличаться от представленных, так как в экспериментах используются случайные числа.

```
BER =
   0.1086
BER_theory =
   0.1029
```

Задание 7. Выполните эксперименты, описанные в примерах 9 и 10 данной работы, а затем промодулируйте и демодулируйте сигналы с дифференциальной фазовой манипуляцией 2 – 3 раза при разных значениях частоты модулирующего и несущего сигналов.

3.9. Частотная модуляция, реализуемая функцией fmmod

Синтаксис функции fmmod описывается соотношениями

>>y = fmmod(x,Fc,Fs,freqdev)

>>y = fmmod(x,Fc,Fs,freqdev,ini_phase)

Функция y = fmmod(x,Fc,Fs,freqdev) модулирует несущую частоту сигналом сообщения x, используя частотную модуляцию. Несущая частота Fc (Гц) и частота дискретизации Fs (Гц), где Fs должно быть по крайней мере равно

2*Fc. Аргумент параметра freqdev представляет собой девиацию частоты, отражаемую постоянным параметром в модулированном сигнале.

Функция y = fmmod(x,Fc,Fs,freqdev,ini_phase) характеризует еще и начальную фазу модулированного сигнала, измеряемую в радианах.

3.10. Частотная демодуляция с помощью функции fmdemod

Функция fmdemod предназначена для выполнения демодуляции сигнала сообщения путем демодулирования модулированного с помощью функции fmmod колебания с частотной модуляцией.

Синтаксис функции fmdemod имеет два варианта задания этой функции:

>>z = fmdemod(y,Fc,Fs,freqdev)

>>z = fmdemod(y,Fc,Fs,freqdev,ini_phase).

Опишем параметры этой функции и покажем, как ее можно использовать для демодуляции высокочастотного колебания с частотной модуляцией, полученной с помощью функции fmmod

Функция z = fmdemod(y,Fc,Fs,freqdev) демодулирует модулированный сигнал, содержащий в своей структуре модулирующий сигнал сообщения *x*, выполняя процедуру частотной демодуляции. Несущая частота сигнала выбирается равной Fc (Гц) и частота дискретизации равна Fs (Гц), где Fs должна равняться по крайней мере не менее чем 2*Fc. Параметр аргумента freqdev представляет собой девиацию частоты и измеряется в герцах и имеет то же значение, которое он содержал во время получения частотно-модулированного сигнала y, полученного с помощью функции fmmod.

Функция z = fmdemod(y,Fc,Fs,freqdev,ini_phase), кроме того, что делает функцию z = fmdemod(y,Fc,Fs,freqdev), учитывает еще начальную фазу модулированного сигнала, измеряемую в радианах.

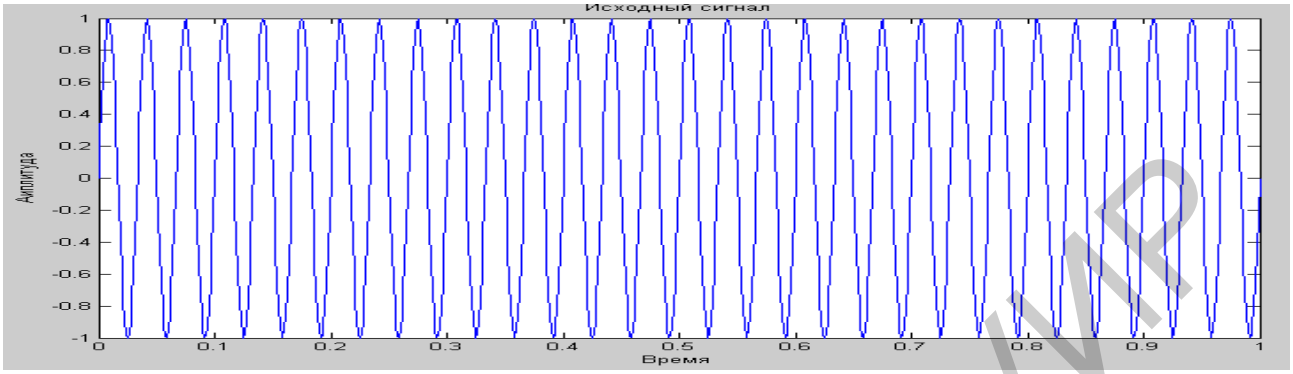
Пример 12. Записать коды, которые модулируют многоканальный (в частности, двухканальный) сигнал, используя функцию модуляции fmmod, и демодулирует его с помощью функции демодуляции fmdemod.

Решение:

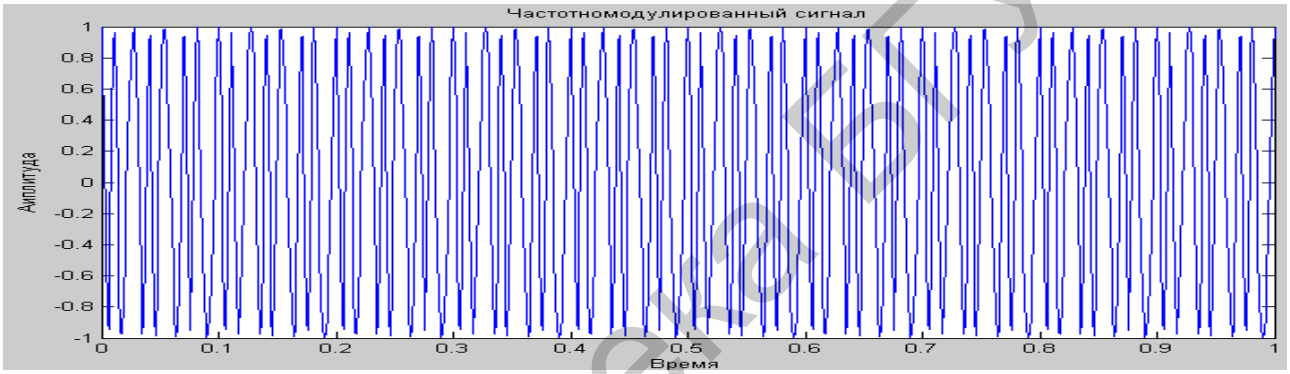
```
>>Fs = 800; % частота дискретизации сигнала
>>Fc = 70; % несущая частота
>>t = [0:Fs]'/Fs; % моменты отсчетов сигнала
>>dev = 30; % Частота девиации модулированного сигнала
>> s1 = sin(2*pi*30*t);
>> figure(1);
>> plot (t, s1);
>> xlabel('Время');
>> ylabel('Амплитуда');
>> title('Исходный сигнал');
>> y = fmmod(s1, Fc, Fs, dev); % модулированный по частоте сигнал
>> figure(2);
>> plot (t, y);
>> xlabel('Время');
>> ylabel('Амплитуда');
>> title('Частотно-модулированный сигнал');
>> z = fmdemod(y, Fc, Fs, dev); %Демодулирование сигнала
>> figure(3);
```

```
>> plot (t, z);
>> xlabel('Время');
>> ylabel('Амплитуда');
>> title('Восстановленный сигнал');
```

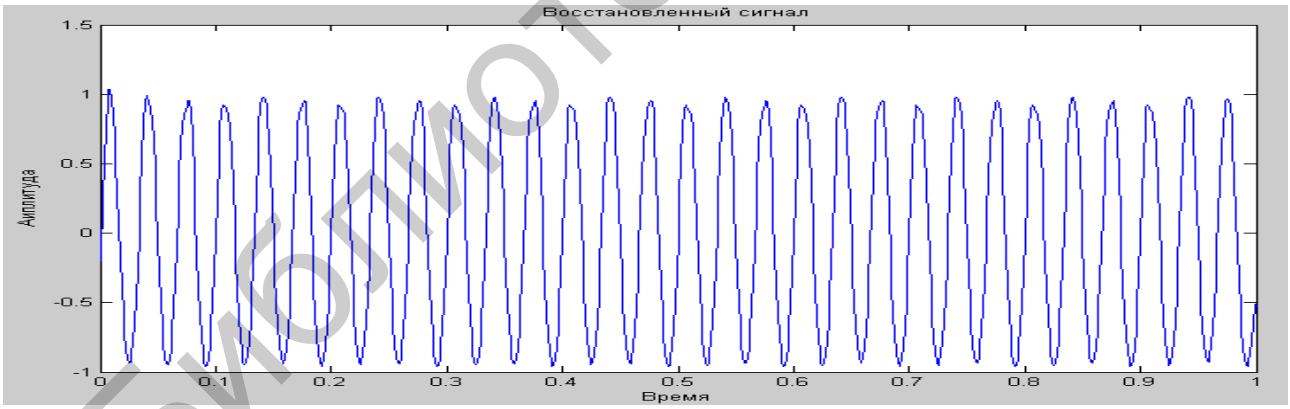
На рис. 12 приведены временные диаграммы исходного, модулированного по частоте и демодулированного сигналов.



а



б



в

Рис. 12. Временные диаграммы исходного сигнала (а), модулированного по частоте сигнала (б) и восстановленного сигнала после выполнения процесса демодуляции (в)

Пример 12. С помощью функций fmmod и fmdemod можно обрабатывать и многоканальные сигналы. Ниже приведен код MATLAB для обработки такого сигнала:

```
>>Fs = 8000; % частота дискретизации сигнала
>>Fc = 3000; % несущая частота
>>t = [0:Fs]'/Fs; % моменты отсчетов сигнала
```

```
>>dev = 50; % Частота девиации модулированного сигнала

>>s1 = sin(2*pi*300*t)+2*sin(2*pi*600*t); % Канал 1
>>s2 = sin(2*pi*150*t)+2*sin(2*pi*900*t); % Канал 2
>>x = [s1,s2]; % Двухканальный сигнал
>>dev = 50; % Девиация частоты в модулированном сигнале
>>y = fmmod(x,Fc,Fs,dev); % Модулируем оба сигнала.
>>z = fmdemod(y,Fc,Fs,dev); % демодулируем оба канала.
```

Задание 8. Выполните эксперименты, описанные в примерах 11 и 12 данной работы, а затем промодулируйте и демодулируйте сигнал с частотной манипуляцией 2 – 3 раза при разных значениях частоты модулирующего и несущего сигналов. Выведите на экран монитора временные диаграммы полученных сигналов.

4. Вопросы для проверки и контроля усвоения материала

1. Какой процесс понимается под угловой модуляцией?
2. Что такое фазовая модуляция и чем она отличается от фазовой манипуляции?
3. Что такое частотная модуляция и чем она отличается от частотной манипуляции?
4. Какие созвездия сигналов с фазовой манипуляцией вам известны?
5. Что представляет собой параметр аргумента freqfev и в каких единицах он измеряется?
6. Какие сигналы можно обрабатывать с помощью функций fmmod и fmdemod?
7. Как функция z = fskdemod(y,M,freq_sep,nsamp) демодулирует комплексную огибающую модулированного сигнал y, полученного методом частотной манипуляции?
8. Какая функция характеризует скорость дискретизации модулированного сигнала *y* в герцах?
9. Какая функция выводит комплексную огибающую модулированного по частоте сообщением *x* сигнала *y*, используя частотную манипуляцию?
10. Какая функция демодулирует комплексную огибающую *y* модулированного сигнала с фазовой манипуляцией?
11. Какая функция возвращает конечную фазу модулированного сигнала *y*?
12. Какая функция характеризует поворот фазы при модуляции, который измеряется в радианах?
13. Какая функция возвращает конечную фазу сигнала *y*, которая имеет важное значение для демодуляции сигналов в будущем?
14. Какая функция демодулирует комплексную огибающую сигнала, использующую метод дифференциальной кодированной частотной модуляции с минимальным сдвигом?

СОДЕРЖАНИЕ

Св. план 2015, поз. 43

Учебное издание

Першин Виктор Тихонович

ФОРМИРОВАНИЕ И ГЕНЕРИРОВАНИЕ СИГНАЛОВ ЦИФРОВОЙ РАДИОСВЯЗИ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

В двух частях

Часть 1

В авторской редакции

Корректор *Е. Н. Батурчик*
Компьютерная правка и оригинал-макет *А. А. Лущикова*

Подписано в печать 12.09.2016. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 11,86. Уч.-изд. л. 12,5. Тираж 50 экз. Заказ 49.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6