Посвящается 50-летию ПГАТИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра ТОРС

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ПЕРЕДАЧИ И ПРИЁМА ДВОИЧНЫХ СИГНАЛОВ В КАНАЛЕ С ГАУССОВСКИМ ШУМОМ С ИСПОЛЬЗОВАНИЕМ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ НА С++

методическая разработка к практическим занятиям для студентов дневной формы обучения по дисциплине «Объектно-ориентированное программирование на С++»

Составители:
доц. Алышев Ю. В.
доц. Борисенков А. В.

Редактор:
проф. Николаев Б. И.

Рецензент
проф. Кораблин М. А.

САМАРА
2006

Компьютерное моделирование передачи и приёма двоичных сигналов в канале с гауссовским шумом с использованием объектно-ориентированного программирования на C++: методическая разработка / Сост. Ю. В. Алышев, А. В. Борисенков. – Самара: ПГАТИ, 2006. – 41 с., ил.

Методическая разработка предназначена для выполнения практических работ по дисциплине «Объектно-ориентированное программирование на C++» студентами дневного отделения специальностей 210302, 210402 и 210405. Содержит задания по внесению изменений и добавлению новых фрагментов в тестовые примеры программ. Итогом является составление программы для статистического моделирования передачи и оптимального приема по критерию максимального правдоподобия двоичных сигналов, а также проверка достоверности моделирования с помощью доверительных интервалов.

Печатается по решению Методического Совета ПГАТИ

Составители: Алышев Ю. В., доцент
Борисенков А. В., доцент

Редактор: Николаев Б. И., профессор

Рецензент: Кораблин М. А., профессор

ВВЕДЕНИЕ

В.1 Модель цифровой системы связи

Передачу цифрового сообщения можно рассматривать в виде абстрактной модели, как показано на рис. 1.

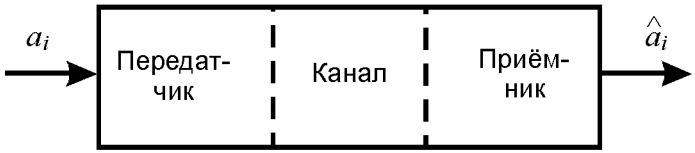

Рис. 1 – Абстрактная модель цифровой системы связи

Здесь a_i – сообщение на передаче, $\hat{a}_i$ – оценка сообщения на приёме, i – индекс, соответствующий отдельному сообщению (символу) и характеризующий его порядковый номер во временной последовательности.

Задачей передатчика является преобразование сообщений в сигналы, с помощью которых эти сообщения можно передавать по каналу связи. В приёмнике делается обратное преобразование и на его выходе ожидается последовательность переданных сообщений. В реальных условиях в канале связи сигналы искажаются под воздействием различных помех. Поэтому на приёмной стороне производится оценивание сигналов. Приёмное устройство, производящее оценивание, может вынести неправильные решения. Как часто приёмное устройство может выносить неправильное решение? Для ответа на этот вопрос необходим анализ рассматриваемой системы связи [1, 2].

В.2 Анализ качества цифровой системы связи

Рассмотрим подробнее процесс передачи цифрового сообщения. Для передачи по каналам связи цифровое сообщение преобразуют в сигнал. Для того, чтобы этот сигнал передать по радиолинии, необходимо, чтобы его спектр соответствовал полосе пропускания канала. Спектр сигнала должен быть ограничен практически, то есть иметь конечную ширину. Далее этот сигнал (его спектр) необходимо перенести в область частот, на которой целесообразно передавать его через радиоканал. На приёмной стороне происходит обратное преобразование спектра сигнала. Далее производится анализ принятого колебания, в результате которого выносится решение в пользу символа, наиболее подходящего по заданному критерию качества.

При переносе спектра сигнала вверх на передающей стороне и вниз на приёмной в устройствах происходят линейные преобразования, которые можно не учитывать при постановке задачи анализа качества полученной оценки $\hat{a}_i$. Анализ качества оценки $\hat{a}_i$ можно производить на базе и математической, и компьютерной модели. Но искажения спектра сигнала в радиоканале можно моделировать и не производя переноса его в верхнюю область частот. Для этого, в простейшем случае, искажения в канале могут быть получены с помощью добавления гауссовского шума. Модель системы связи, содержащей канал с аддитивным гауссовским шумом, показана на рис. 2.

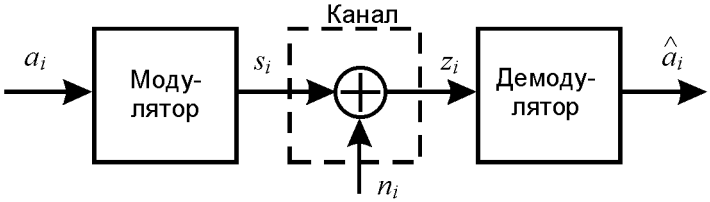

Рис. 2 – Модель системы связи, содержащей канал с аддитивным гауссовским шумом

Здесь рассмотрены следующие блоки:

модулятор – устройство, в котором параметр переносчика (несущей) меняется по закону первичного (входного) сигнала;

демодулятор – устройство, которое выносит решение в пользу той или иной гипотезы при анализе сигнальной реализации, искажённой в канале связи.

Под гипотезами понимаются символы, которые были использованы в качестве информационного сообщения на передаче. Количество гипотез обычно соответствует количеству символов на передающей стороне. В простейшем случае рассматривают передачу двоичных символов (бит).

В реальных устройствах связи сигнал на выходе модулятора и на входе демодулятора аналоговый. Однако при моделировании этих устройств на ЭВМ необходимо перейти к дискретному или цифровому сигналу. С учётом того, что над спектром сигнала не производится преобразование вверх и, в соответствии с теоремой Котельникова, сигнал на выходе модулятора можно представить в виде последовательности отсчётов с заданной частотой дискретизации. Конкретную частоту дискретизации выбирают из ряда стандартизованных частот и являющейся наиболее эффективной для поставленной задачи. В рамках поставленной задачи гауссовский шум также можно представить в виде

случайного сигнала, дискретные значения которого имеют гауссовское распределение и следуют с той же частотой дискретизации.

Для систем связи, передающих цифровую информацию, одной из качественных характеристик является *вероятность ошибки*. Для многих видов модуляции и демодуляции, выведены теоретические значения вероятности ошибки. При моделировании на ЭВМ понятию «вероятность ошибки» соответствует понятие *частость ошибки*, показывающая насколько часто появляются ошибки в текущем сеансе связи. Частость ошибки определяется как

$$p'_{\text{ош}} = \frac{n_{\text{ош}}}{N} \qquad (1)$$

где $n_{\text{ош}}$ – число ошибок, которые произошли за текущий сеанс связи при передаче N символов.

Из (1) видно, что частость ошибки можно определить, лишь зная число ошибок в текущем сеансе связи. Поэтому число ошибок $n_{\text{ош}}$ является первичным искомым параметром для процесса моделирования.

В.3 Модель для статистических испытаний

При моделировании можно получить ряд случайных искомых значений параметра $n_{\text{ош}}$. В рамках теории математической статистики речь идёт о проведении статистических испытаний. Для проведения статистических испытаний должна быть построена соответствующая модель. Эту модель эффективно реализуется на компьютере.

Модель для статистических испытаний должна содержать источник сигнала, выдающий двоичные символы, модель модулятора, модель канала связи, вносящего искажения в сигнал, модель демодулятора и счётчик ошибок.

Для реализации данной модели на компьютере предлагается воспользоваться наиболее распространённым инструментом для написания технических программ, таким как языки C [5] и C++ [6]. Мотивацией в пользу этого инструмента является то, что устройства обработки цифровых сигналов делаются чаще всего на базе сигнальных процессоров. При этом программное обеспечение для сигнальных процессоров содержит два языка программирования: ассемблер и язык C/C++. При этом последний является универсальным инструментом, который имеет единую форму для большого количества сигнальных процессоров, которые могут иметь разную производственную платформу.

В.4 Подготовка рабочего места и знакомство с редактором C++

В.4.1 Создание каталога и файла в Norton (Volkov) Commander

1 Найдите каталог `USERS` (этот каталог может находиться на других дисках; переход на другие диски осуществляется клавишами «`Alt`»+«`F1`» или «`Alt`»+«`F2`»). Для входа в любой каталог переведите курсор с помощью клавиш «↑», «↓» на него и нажмите «`Enter`». Для выхода из каталога курсор необходимо перевести на самый верх, где находятся две точки «`..`» и нажать «`Enter`».

2 В каталоге `USERS` создайте свой подкаталог (для этого воспользуйтесь функциональной клавишей «`F7`»). После этого возникнет диалоговое окно, в котором введите имя подкаталога латинскими буквами (не более 8 символов), и нажмите клавишу «`Enter`». Установите курсор на этот подкаталог. Войдите в него, нажав «`Enter`».

3 Чтобы создать файл, воспользуйтесь комбинацией клавиш «`Shift`»+«`F4`». После этого появится диалоговое окно. Введите имя файла (латинскими буквами не более 8 символов), затем через точку расширение «`cpp`», например «`program.cpp`».

4 Если появится сообщение «`Can't find the file`», то в ответ на это нажмите клавишу «`Enter`».

5 На экране монитора появится простейший текстовый редактор. Нажмите клавишу «**`F2`**» (сохранение).

6 В результате в Ваш каталог будет записан созданный вами файл.

7 Чтобы убедиться, что файл создан, выйдите из редактора, нажав клавишу «**`F10`**» или «**`ESC`**». В Вашем каталоге должен появиться новый файл. Если он не появился, то выполните повторно действия, начиная с п. 3.

8 Для загрузки созданного файла в редактор языка C++ переведите курсор на созданный файл и нажмите клавишу «`Enter`».

В.4.2 Начальное обучение в редакторе языка C++ фирмы Borland

1 Вверху экрана расположено меню редактора. Для входа в меню используется клавиша «**`F10`**» (или сразу для входа в выпадающее подменю – комбинация клавиш «`Alt`» плюс подсвеченная буква в пункте меню, тогда п. 2 можно пропустить).

2 После нажатия клавиши «**`F10`**» один из пунктов меню подсветится зелёным цветом. С помощью стрелок «←» или «→» можно перемещаться в другие пункты меню. При нажатии клавиши «↓» или «`Enter`» появляется выпадающее подменю.

3 Обучение созданию, открытию и закрытию файла.

Рассмотрим некоторые пункты подменю «File»:

– «New» – создание нового файла.

– «Open» – открытие файла. При этом появляется окно, где из списка выбирается нужный файл. Для перехода используются клавиши «Tab», «Shift+Tab», «↑», «↓», «←», «→», для подтверждения – «Enter», для отмены – «Esc». Эти клавиши действуют и в других диалоговых окнах.

– «Save» – сохранение текущего файла.

– «Save as...» – для сохранения информации в файле с новым именем. Это необходимо для создания копии файла, например, для того, чтобы не набивать заново данные при написании другой программы.

– «Quit» – выход из редактора.

3.1 Создайте новый файл, используя пункт «New». Появится окно с названием «NONAME00.CPP» (это означает, что файл без имени).

3.2 Сохраните его (пункт «Save»). При этом появится окно, в котором необходимо указать имя файла (например, «prog1.cpp»).

После сохранения наберите некоторый текст на латинском шрифте в текущем окне. Обратите внимание на нижний левый угол текущего окна. Слева от цифр, разделённых двоеточием, которые обозначают номер строки и номер столбца, возникнет звёздочка «*».

Перейдите в подменю «File» к пункту «Save» (горячая клавиша «**F2**») и нажмите «**Enter**». Звёздочка исчезнет. Если продолжите что-нибудь набирать, звёздочка опять возникнет и пропадёт при нажатии клавиши «**F2**», нажимая клавишу «**F2**», Вы сохраняете набранный файл на жёстком диске. **Необходимо, чтобы нажатие клавиши «F2» вошло в привычку**, особенно перед компиляцией Вашей программы.

3.3 Сохраните этот файл под другим именем в своем подкаталоге, используя пункт «Save as...». В появившемся окне введите имя (например «program2» или «program2.cpp») и нажмите «Enter». Причем, если вы набрали только имя программы, то расширение «.cpp» поставит сам редактор.

3.4 Выйдите из редактора, используя пункт «Quit» в подменю «File» (горячие клавиши «**ALT**»+«**X**»). Убедитесь в создании и сохранении двух файлов (например «prog1.cpp» и «program2.cpp»). Для этого переведите курсор на соответствующий файл и нажмите «F3». На экране появится окно для просмотра файлов, убедитесь в сохранении информации и, для выхода из окна, нажмите клавишу «Esc» или «F10». Переведите курсор на любой из файлов, имеющих расширение «.cpp», и нажмите «Enter», при этом запустится редактор.

Войдите в подменю «File» и перейдите к пункту «Open» (горячая клавиша «**F3**»). С помощью клавиши «Tab» и «↓» выберите в пункте «Files» нужный файл. Нажмите «Enter». Таким образом, вы загрузите выбранный вами файл в новое окно.

4 Обучение работы с окном.

Рассмотрим некоторые пункты подменю «Window» (окно):

– «Size/Move» – перемещение и изменение размера активного окна.

– «Zoom» – разворачивание окна на весь экран и в исходное положение.

– «Tile» и «Cascade» предназначены для расположения открытых окон на экране в различном виде.

– «Next» – перемещение из одного окна в другое.

– «Close» и «Close all» – для закрытия текущего окна или всех окон.

4.1 Перейдите в подменю «Window» к пункту «Size/Move» (горячая клавиша «**Ctrl**»+«**F5**») и нажмите «Enter». Текущее окно будет иметь рамку из одинарной линии. В таком режиме с помощью стрелок, «←», «↑», «→» и «↓» можно перемещать текущее окно. А с помощью комбинаций «Shift» и стрелок – изменять размеры окна. Клавишей «Enter» можно закрепить текущее состояние окна, при этом рамка окна будет иметь двойную линию.

4.2 Перейдите в подменю «Window» и перейдите к пункту «Zoom» (горячая клавиша «**F5**»), нажмите «Enter». При этом окно развернётся на весь экран. При повторном нажатии вернётся в исходное состояние. Нажмите несколько раз клавишу «F5».

4.3 Перейдите в подменю «Window» к пункту «Tile», нажмите «Enter». Перейдите в подменю «Window» и перейдите к пункту «Cascade», нажмите «Enter». Проделайте последние две инструкции несколько раз. В форме «Cascade» есть полные названия заголовков, а в форме «Tile» все окна расположены на экране так, чтобы занять всю площадь рабочей панели.

4.4 Перейдите в подменю «Window» к пункту «Tile», нажмите «Enter». Перейдите в подменю «Window» и перейдите к пункту «Next» (горячая клавиша «F6»), нажмите «Enter». Нажмите несколько раз клавишу «F6». Таким образом, по кругу осуществляется переход между окнами.

4.5 Перейдите в подменю «Window» к пункту «Close» (горячая клавиша «Alt»+«F3»), нажмите «Enter». Текущее окно закроется. Таким образом, с помощью клавиш «Alt»+«F3» можно закрывать окна. Откройте 2 или более файлов, если они не открыты. Перейдите в подменю «Window» к пункту «Close all». После нажатия «Enter» убедитесь в закрытии всех окон.

5 Обучение быстрому набору и редактированию текстов программ.

Рассмотрим некоторые пункты подменю «Edit» (редактировать):

– «Undo» – отменить последнее редактирование.

– «Redo» – отменить отмену последнего редактирования.

– «Cut» – вырезать выделенный серым цветом текст.

– «Copy» – копировать выделенный серым цветом текст.

– «Paste» вставить ранее скопированный текст.

– «Clear» – очистить выделенный серым цветом текст.

– «Show clipboard» – просмотреть содержимое буфера памяти.

5.1 Создайте новый файл, руководствуясь вышеизложенной информацией. Наберите некоторый текст, состоящий из нескольких строк (переход на следующую строку – клавиша «Enter»).

Переведите курсор на первую строку (клавиша «↑»). Нажмите клавишу «End» (курсор установится в конце строки). Нажмите клавишу «Home» (курсор установится в начале строки).

5.2 Нажмите клавишу «Shift» и, удерживая её в нажатом положении, нажмите 2 раза клавишу «↓», при этом текст должен выделиться серым цветом. Выделение фрагмента осуществляется с помощью клавиш «←», «↑», «→» и «↓» при одновременном удерживании клавиши «Shift».

5.3 Нажмите «F10». Войдите в подменю «Edit» и перейдите к пункту «Copy». Нажмите «Enter». Выделенный текст копируется в буферную память: «карман», то же самое можно выполнить с помощью комбинации клавиш «Ctrl»+«Ins».

5.4 Переведите курсор в конец файла с помощью клавиши «↓».

5.5 Нажмите «F10». Войдите в подменю «Edit» и перейдите к пункту «Paste». Нажмите «Enter». Выделенный текст копируется из буферной памяти по месту курсора, то же самое можно выполнить с помощью комбинации клавиш «Shift»+«Ins».

5.6 Пункт 5.5. можно повторить несколько раз. При этом столько же будет создано копий выделенного фрагмента.

5.7 Выделите некоторый фрагмент Вашего текста.

5.8 Нажмите «F10». Войдите в подменю «Edit» и перейдите к пункту «Cut». Нажмите «Enter». Выделенный текст переносится в буферную память: «clipboard», при этом выделенная область в тексте пропадает, то же самое можно выполнить с помощью комбинации клавиш «Shift»+«Del».

5.9 Повторите п. 5.7. Далее нажмите «F10». Войдите в подменю «Edit» и перейдите к пункту «Clear». Нажмите «Enter». Выделенная область в тексте пропадает, то же самое можно выполнить с помощью комбинации клавиш «Ctrl»+«Del».

5.10 Далее, можно выполнить п.п. 5.4, 5.5 и 5.6. При этом текст будет вставлен из буферной памяти.

5.11 То, что копируется из буферной памяти – выделенный фрагмент в окне, которое открывается следующими нажатиями клавиш: нажмите «F10», войдите в подменю «Edit» и перейдите к пункту «Show Clipboard», нажмите «Enter». В конце файла модно найти выделенный фрагмент. После просмотра буферной памяти его можно закрыть.

5.12 Нажмите «F10». Войдите в подменю «Edit» и перейдите к пункту «Undo». Нажмите «Enter». (Горячие клавиши «Alt»+«Backspace»).

5.13 Выполните п. 5.12 несколько раз. Желательно нажимать «горячие клавиши», при этом на экране будут постепенно отменяться все действия, которые Вы перед этим делали.

5.14 Нажмите «F10». Войдите в подменю «Edit» и перейдите к пункту «Redo». Нажмите «Enter». (Горячие клавиши «Shift»+«Alt»+«Backspace»).

5.15 Выполните п. 5.14 несколько раз. Желательно нажимать «горячие клавиши», при этом на экране будут постепенно восстанавливаться все действия, которые Вы перед этим отменили.

6 Обучение быстрому поиску и замене текстов программ.

Рассмотрим некоторые пункты подменю «Search» (редактировать):

– «Find...» – поиск текста.

– «Replace...» – замена текста на ....

– «Search again» поиск следующей строки.

6.1 Переведите курсор в начало текста. Нажмите «F10». Войдите в подменю «Search» и перейдите к пункту «Find...». Нажмите «Enter». На экране появится диалоговое окно для поиска строки. Наберите строку для поиска (желательно, чтобы эта строка существовала в вашем тексте) и нажмите «Enter». Редактор выделит найденную строку и остановит за ней курсор.

6.2 Нажмите «F10». Войдите в подменю «Search» и перейдите к пункту «Search again». (Горячие клавиши «Ctrl»+«L») Редактор выделит следующую найденную строку и остановит за ней курсор или выдаст сообщение «Search string not found» («Строка не найдена»).

6.3 Переведите курсор в начало текста. Нажмите «F10». Войдите в подменю «Search» и перейдите к пункту «Replace...». Нажмите «Enter». На экране появится диалоговое окно для поиска символьной последовательности и замены её на другую. Наберите строку для поиска (желательно, чтобы эта строка существовала в Вашем тексте) перейдите с помощью клавиши «Tab» в другое окно, наберите текст, для замены и нажмите «Enter». На экране появится вопрос: «Replace this

occurrence?» и варианты ответа «Yes» (Горячая клавиша «Y», заменить первое найденное), «Change all» (Горячая клавиша «A», заменить всё, что есть в тексте), «No» (Горячая клавиша «N», не менять), «Cancel». Клавиша «Esc», отменить замену. Выбрать ответ можно с помощью клавиш «Tab» или «Shift»+«Tab» и нажать «Enter» или с помощью горячих клавиш. Редактор заменит найденную строку на новую, выделит её, и остановит за ней курсор.

7 Создание первой программы.

Создайте новый файл. Наберите нижеприведённый текст.

```
#include <stdio.h>
void main()
{
printf("Привет!\n");
}
```

Это Ваша первая программа, выводящая на экран «Привет!».

7.1 Нажмите «F10». Войдите в подменю «Run» и перейдите к пункту «Run». (Горячие клавиши «Ctrl»+«F9»). Произойдёт компиляция и запуск программы на выполнение.

7.2 Если на экране появится окно «Compiling» с мигающим сообщением «Press any key», то это означает, что в процессе набора текста вы допустили ошибку. Чтобы научиться выявлять ошибки в программе, позовите преподавателя.

7.3 Нажмите «F10». Войдите в подменю «Window» и перейдите к пункту «User screen» (Горячие клавиши «Alt»+«F5»). На чёрном экране внизу должно появиться слово «Привет!». Возврат в редактор – клавиша «Esc». Данная программа написана в стиле языка С.

8 Создайте новый файл. Наберите нижеприведённый текст.

```
#include <iostream.h>
void main()
{
cout<< "Здравствуйте!"<<endl;
}
```

8.1 Выполните п.п. 7.1 и7.2.

8.2 Нажмите «F10». Войдите в подменю «Window» и перейдите к пункту «User screen» (Горячие клавиши «Alt»+«F5»). На чёрном экране внизу должно появиться слово «Здравствуйте!». Возврат в редактор – клавиша «Esc». Данная программа написана в стиле языка С++.

ОПТИМАЛЬНАЯ ДЕМОДУЛЯЦИЯ ДВОИЧНЫХ СИГНАЛОВ

Цель работы: моделированием на языке С++ найти вероятность ошибочного приёма при оптимальном когерентном приёме двоичных сообщений с заданными априорными вероятностями в идеальном канале с гауссовским шумом.

Модель системы передачи двоичных сообщений представляет программу, составленную на языке С++ с использованием объектно-ориентированного программирования. Прежде чем воспользоваться данной программой, необходимо выполнить некоторые задания для ознакомления с методикой моделирования сигналов и устройств. Все задания построены следующим образом: от анализа простых примеров программ происходит постепенный переход к анализу более сложных, и впоследствии к анализу программы, моделирующей систему передачи двоичных сообщений.

1 Дискретизация сигналов

Представить заданный сигнал $A(t)$, $t \in [0,T]$ в виде дискретных отсчётов, если частота дискретизации равна $f_д = 8$ кГц. Сигнал и его длительность выбирается из табл. 1 по формуле

$$\text{№ варианта} = (\text{№ бригады} + \text{№ группы}) \bmod 8 + 1,$$

где «mod 8» – операция деления по модулю 8. Номер группы спросить у преподавателя.

Таблица 1 – Номер варианта, форма сигнала и длительность сигнала

№	$A(t)$	T, мс
1	$\sin(\pi t/T)$	1
2	$\sin^2(\pi t/T)$	1,5
3	$\exp\left(-(t-T/2)^2/2\sigma^2\right)$, $\sigma = T/6$	1,5
4	$\frac{1}{2} - \left\lvert \frac{t}{T} - \frac{1}{2} \right\rvert$	1
5	$\sin(\pi t/T)$	1,5
6	$\sin^2(\pi t/T)$	2
7	$\exp\left(-(t-T/2)^2/2\sigma^2\right)$, $\sigma = T/6$	2
8	$\frac{1}{2} - \left\lvert \frac{t}{T} - \frac{1}{2} \right\rvert$	1,25

1.1 Отображение сигнала вывод сигнала в консоль

Вывести значения отсчётов сигнала в консольное устройство. Записать в тетрадь значения отсчётов. При использовании математических функций в начале программы с помощью команды `#include` необходимо подключить математическую библиотеку `<math.h>` (см. прил.).

Пример

Задан сигнал пилообразной формы:

$$A(t) = t/T\ ;\ t \in [0,T]\ \text{и}\ T = 0,625\ \text{мс}\ .$$

За время длительности сигнала T число дискретных отсчётов составит

$$n = f_{\text{д}} \cdot T = 8000\ \text{Гц} \cdot 0,000625\ \text{с} = 5\ .$$

Ниже приведена программа на языке С++, которая выводит значения отсчётов сигнала на экран в символьном виде

```
#include<iostream.h>
#include<conio.h>
void main()
{
int i;
int mt=5;
for(i=0;i<mt;i++)
 cout<<"s["<<i<<"]="<<(i+0.5)/mt<<endl;
getch();
}
```

1.2 Вывод значений сигнала в файл

Записать значения отсчётов сигнала в файл в текстовом виде. Сравнить значения этих отсчётов с отсчётами, записанными в тетрадь в п. 1.1.

Пример

Ниже приведена программа на языке С++, для сигнала, заданного в примере п. 1.1, которая записывает значения отсчётов сигнала в файл в текстовом виде

```
#include<fstream.h>
#include<stdlib.h>
#include<conio.h>
void main()
```

```
{
ofstream  out("output.dat");
if(!out)
 cerr<<"Файл \"output.dat\" не создан!\n",exit(1);
int i;
int mt=5;
out<<"Номер отсчета   Значение отсчета"<<endl;
for(i=0;i<mt;i++)
 out<<i<<"\t\t"<<(i+0.5)/mt<<endl;
out.close();
cout<<"Результаты занесены в файл \"output.dat\"\n";
getch();
}
```

1.3 Графическая иллюстрация процесса дискретизации

Показать процесс дискретизации с помощью двух графических окон, расположенных друг под другом. В верхнем окне отобразить аналоговый сигнал $A(t)$. В нижнем окне отобразить дискретный во времени сигнал A_i. Зарисовать в тетрадь результат вывода в графические окна.

Пример

Ниже приведена программа на языке C++, для сигнала, заданного в примере п. 1.1. Для графического отображения сигналов используется класс «*osc*» (см. прил.). В верхнем окне с помощью функции «*dx*» рисуется непрерывный сигнал. В нижнем окне с помощью функции «*dxl*» рисуется дискретный сигнал. Необходимо набрать программу, отладить, если допущены ошибки при наборе, и запустить на выполнение.

```
#include<iostream.h>
#include<oscgraph.h>
#include<conio.h>
void main()
{
int i;
int mt=5;
opengraph();
```

```
osc  window1(25,1,75,49);  window1.rect(1);
window1.assignMy(100);
osc  window2(25,51,75,99);  window2.rect(1);
window2.assignMy(100);
window2.step(50);  setcolor(12);
window1.PutMess("Аналоговый сигнал");
window2.PutMess("Дискретный во времени сигнал");
for(i=0;i<25;i++)
 window1.dx(0);
for(i=0;i<mt*50+1;i++)
 window1.dx((i+0.5)/mt/50);
for(i=0;i<25;i++)
 window1.dx(0);
for(i=0;i<mt;i++)
 window2.dxl((i+0.5)/mt);
window2.dxl(0);
getch();
closegraph();
}
```

(примечание: здесь и далее функция «*.dxl*», а не «*.dx*1»!!!)

2 Использование динамического массива для хранения отсчётов сигнала

Записать значения отсчётов дискретизированного финитного сигнала (из п. 1) в динамический массив. Динамический массив описывается классом с шаблонной подстановкой `vect` (см. прил.). Вывести значения отсчётов сигнала из массива на экран в символьном виде. Найти энергию рассматриваемого финитного сигнала.

Энергия сигнала определяется формулой

$$E = \int_0^T A^2(t)dt .$$

Записать в тетрадь значение энергии финитного сигнала.

Пример

Ниже приведена программа на языке C++, для сигнала, заданного в примере п. 1.1. Сначала производится запись сигнала в массив, затем – вывод отсчётов сигнала из массива на экран. Необходимо набрать программу, отладить, если допущены ошибки при наборе, и запустить на выполнение.

```
#include<iostream.h>
#include<vect.h>
void main()
{
int i;
int mt=5;
vect<float> s(mt);
for(i=0;i<mt;i++)
 s[i]=(i+0.5)/mt;
for(i=0;i<mt;i++)
 cout<<"s["<<i<<"]="<<s[i]<<endl;
}
```

Примечание: в данном примере отсутствует фрагмент программы нахождения энергии сигнала, этот фрагмент добавляется в программу самостоятельно.

3 Комплексное представление сигнала

Используя класс комплексных чисел «compl», представить два двоичных комплексных сигнала в виде дискретных отсчётов.

$$\dot{s}_1(t) = A(t)\dot{u}_1(t), \quad \dot{s}_2(t) = A(t)\dot{u}_2(t),$$

где $A(t)$ – огибающая сигнала, которая определяется по вариантам из таблицы 1.

Сигнал $\dot{u}_j(t)$ определяется согласно заданному виду модуляции (табл. 2).

Таблица 2 – Варианты видов модуляции

№ варианта	Вид модуляции
1	ЧМ2: $f_1 = \frac{1}{T}$, $f_2 = \frac{2}{T}$
2	АМ2
3	ЧМ2: $f_1 = 0$, $f_2 = \frac{2}{T}$
4	ФМ2
5	ЧМ2: $f_1 = -\frac{1}{T}$, $f_2 = \frac{1}{T}$
6	МОС
7	ЧМ2: $f_1 = -\frac{1}{T}$, $f_2 = \frac{2}{T}$

Номер варианта в табл. 2 рассчитывается по формуле: $N\%7+1$, где N – порядковый номер студента в списке группы, а «%» – операция «взятие остатка от деления».

В таблице определены следующие виды модуляции: частотная модуляция двоичными сигналами – ЧМ2, амплитудная модуляция двоичными сигналами (амплитудная манипуляция) – АМ2, фазовая модуляция двоичными сигналами (модуляция противоположными сигналами) – ФМ2, модуляция ортогональными сигналами – МОС.

В случае ЧМ2:

$$\dot{u}_j(t) = \mathrm{e}^{\mathrm{j}2\pi(f_0+f_j)t}, \quad f_j = \begin{cases} f_1, & j=1, \\ f_2, & j=2, \end{cases} \tag{1}$$

где f_1 и f_2 заданы в табл. 2.

В случае АМ2:

$$\dot{u}_j(t) = a_j \mathrm{e}^{\mathrm{j}2\pi f_0 t}, \quad a_j = \begin{cases} 1, & j=1, \\ 0, & j=2. \end{cases} \tag{2}$$

В случае ФМ2 и МОС:

$$\dot{u}_j(t) = \mathrm{e}^{\mathrm{j}\left(2\pi f_0 t+\varphi_j\right)}, \tag{3}$$

ФМ2: $\varphi_j = \begin{cases} 0, & j=1, \\ \pi, & j=2, \end{cases}$ МОС: $\varphi_j = \begin{cases} 0, & j=1, \\ \pi/2, & j=2. \end{cases}$

В формулах (1), (2) и (3) табл. 2 f_0 – базовая частота сигнала, которая определяется согласно варианту из табл. 3.

Таблица 3 – Варианты значений базовой частоты

№ варианта	1	2	3	4
f_0	$4/T$	$6/T$	$8/T$	$10/T$

Номер варианта из табл. 3 рассчитывается по формуле: $N/7+1$. Здесь используется операция целочисленного деления, которая используется в вычислительной технике. В результате этого деления отбрасывается дробная часть.

При представлении сигнала в виде квадратурных компонент, необходимо в формулах (1), (2) и (3) значение базовой частоты $f_0 = 0$. Это соответствует переносу спектра сигнала на величину f_0 в нижнюю область частот.

Здесь и далее частоту дискретизации выбирать произвольно, не нарушая условия теоремы Котельникова.

3.1 Вывод значений комплексного сигнала в текстовый файл

Вывести значения вещественной и мнимой части дискретизированного финитного комплексного сигнала в текстовый файл.

3.2 Вывод значений комплексного сигнала в бинарный файл

Вывести значения вещественной и мнимой части дискретизированного финитного комплексного сигнала в бинарный файл.

3.3 Графическая иллюстрация комплексного сигнала

Вывести в графические окна значения отсчётов вещественной и мнимой части комплексного финитного сигнала. Отобразить вещественную компоненту X в верхнем окне, а мнимую компоненту Y – в нижнем.

3.4 Вывод значений квадратурных компонент сигнала в текстовый файл

Вывести значения вещественной и мнимой части дискретизированного финитного сигнала в текстовый файл.

3.5 Вывод значений квадратурных компонент сигнала в бинарный файл

Вывести значения вещественной и мнимой части дискретизированного финитного сигнала в бинарный файл.

3.6 Графическая иллюстрация сигнала в виде квадратурных компонент

Вывести значения квадратурных компонент дискретизированного финитного сигнала в графические окна. Отобразить синфазную компоненту X в верхнем окне, а квадратурную компоненту Y – в нижнем.

Пример

Ниже приведена программа на языке C++, для сигнала, заданного в примере п. 1.1. В примере рассмотрен случай квадратурной модуляции (КМ). В каждой компоненте формируется отдельный сигнал: в компоненте X: $s_X(t) = t/T$; в компоненте Y: $s_Y(t) = (T-t)/T$. В ***бинарный файл*** «`output.qfb`» записываются значения отсчётов квадратурных компонент двух сигналов. Первый сигнал соответствует передаче символа «0», а второй – передаче символа «1». Необходимо набрать программу, отладить, если допущены ошибки при наборе, и запустить на выполнение.

```
#include<fstream.h>
#include<conio.h>
#include<vect.h>
#include<quadr.h>
void main()
```

```
{
int i, mt=5;
clrscr();
ofstream out("output.qfb",ios::binary);
if(!out) cerr<<"Файл \"output.qfb\" не
создан!\n",exit(1);
vect<float> sx(mt), sy(mt);
vect<quadr<float> > s0(mt), s1(mt);
for(i=0;i<mt;i++)
 sx[i]=(i+0.5)/mt, sy[i]=(mt-0.5-i)/mt;
for(i=0;i<mt;i++)
 s0[i](-sx[i],-sy[i]), s1[i](sx[i],sy[i]);
for(i=0;i<mt;i++)
 s0[i].write_bin(out);
for(i=0;i<mt;i++)
 s1[i].write_bin(out);
out.close(); getch();
}
```

Просмотр сигнала возможен с помощью специальной программы. Эту программу можно взять у преподавателя.

4 Модель модулятора и генератора информационных символов

Получить последовательность отсчётов квадратурных компонент сигнала заданного вида модуляции. На вход модулятора поступает информационная последовательность двоичных символов. Символы формируются генератором, в котором заданы их априорные вероятности. Ниже приведена таблица численных значений априорных вероятностей символов по вариантам.

Таблица 4 – Значения априорной вероятности информационных символов

№ вар.	$P(0)$	$P(1)$	№ вар.	$P(0)$	$P(1)$
1	0,1 (10%)	0,9 (90%)	7	0,35 (35%)	0,65 (65%)
2	0,85 (85%)	0,15 (15%)	8	0,7 (70%)	0,3 (30%)
3	0,2 (20%)	0,8 (80%)	9	0,25 (25%)	0,75 (75%)
4	0,75 (75%)	0,25 (25%)	10	0,8 (80%)	0,2 (20%)
5	0,3 (30%)	0,7 (70%)	11	0,15 (15%)	0,85 (85%)
6	0,65 (65%)	0,35 (35%)	12	0,9 (90%)	0,1 (10%)

Номер варианта N рассчитывается по формуле $N = (G + B)\ \%\ 12 + 1$, где G – номер группы (спросить у преподавателя), B – номер бригады. Для выполнения следующих заданий воспользуйтесь примером п. 3 и примером в этом пункте. Сигнал на выходе модулятора определяется по вариантам (для каждого студента) согласно табл. 1–3.

4.1 Графическая иллюстрация дискретизированного сигнала на выходе модулятора в виде квадратурных компонент

Вывести значения квадратурных компонент дискретизированного сигнала на выходе модулятора в графические окна. Отобразить синфазную компоненту X в верхнем окне, а квадратурную компоненту Y – в нижнем. Частота дискретизации 8 кГц. Число информационных символов – 15.

4.2 Графическая иллюстрация аналогового сигнала на выходе модулятора в виде квадратурных компонент

Изобразить квадратурные компоненты сигнала на выходе модулятора в графических окнах. Число информационных символов – 10. Для отображения аналогового сигнала воспользуйтесь функцией «*dxl*», при этом частоту дискретизации выбрать самостоятельно согласно теореме Котельникова.

4.3 Графическая иллюстрация дискретизированного комплексного сигнала на выходе модулятора

Вывести значения отсчётов вещественной и мнимой части комплексного сигнала на выходе модулятора в графические окна. Отобразить синфазную компоненту X в верхнем окне, а квадратурную компоненту Y – в нижнем. Частота дискретизации 8 кГц. Число информационных символов – 10.

4.4 Графическая иллюстрация аналогового комплексного сигнала на выходе модулятора

Изобразить вещественную и мнимую часть комплексного сигнала на выходе модулятора в графических окнах. Число информационных символов – 10. Для отображения аналогового сигнала воспользуйтесь функцией «*dxl*», при этом частоту дискретизации выбрать самостоятельно согласно теореме Котельникова.

Пример

В примере рассмотрен случай квадратурной модуляции (КМ). На вход модулятора поступают две информационные последовательности двоичных символов. Символы формируются двумя генераторами. Каждый генератор формирует поток двоичных символов.

$$s_{\mathrm{X}}(t) = \sum_{i=0}^{N-1} s_{b_i,\mathrm{X}}(t - iT), \qquad s_{\mathrm{Y}}(t) = \sum_{i=0}^{N-1} s_{b_i,\mathrm{Y}}(t - iT), \tag{4}$$

где N – число информационных символов для каждой компоненты;

$b_{i,\mathrm{X}}$, $b_{i,\mathrm{Y}}$ – информационные элементы, соответствующие синфазной X и квадратурной Y компонентам сигнала.

В компоненте X: $s_{0\mathrm{X}}(t)=\frac{t}{T}\cos f_1 t$, если 1-й генератор выдаёт символ «0» и $s_{1\mathrm{X}}(t)=\frac{t}{T}\cos f_2 t$ если 1-й генератор выдаёт символ «1». В компоненте Y: $s_{0\mathrm{Y}}(t)=\frac{T-t}{T}\sin f_1 t$ и $s_{1\mathrm{Y}}(t)=\frac{T-t}{T}\sin f_2 t$ соответственно. Для финитных сигналов $t\in[0,T]$.

Вероятность появления каждого из двоичных символов (априорная вероятность) равна 0,5. В каждой компоненте формируется отдельный сигнал, состоящий из финитных сигналов, сдвинутых во времени на соответствующую величину, кратную тактовому интервалу. Необходимо набрать программу, отладить, если допущены ошибки при наборе, и запустить на выполнение.

```
#include<vect.h>
#include<compl.h>
#include<oscgraph.h>
void main()
{
int j,i,mt=60;
float  f1=8*M_PI/mt,f2=16*M_PI/mt;
vect<float> sx(mt),sy(mt);
vect<compl<float> > s0(mt),s1(mt);
for(i=0;i<mt;i++)
 sx[i]=(i+0.5)/mt,  sy[i]=1-(i+0.5)/mt;
for(i=0;i<mt;i++)
 {
 s0[i](sx[i]*cos(f1*i),sy[i]*sin(f1*i));
 s1[i](sx[i]*cos(f2*i),sy[i]*sin(f2*i));
 }
opengraph();  setcolor(12);
osc graphX(1,1,99,49);   graphX.rect(1);
```

```
graphX.assignMy(100);
graphX.PutMess("Осциллограмма компоненты X");
osc graphY(1,51,99,99); graphY.rect(1);
graphY.assignMy(100);
graphY.PutMess("Осциллограмма компоненты Y");
for(j=0;j<20;j++)
 {
 int bitX=random(2); // 1-й генератор
 int bitY=random(2); // 2-й генератор
 for(i=0;i<mt;i++)
  {
  if(!bitX)
   graphX.assignColor(10),graphX.dx(s0[i].x());
  else
   graphX.assignColor(12),graphX.dx(s1[i].x());
  if(!bitY)
   graphY.assignColor(10),graphY.dx(s0[i].y());
  else
   graphY.assignColor(12),graphY.dx(s1[i].y());
  }
 }
getch();
closegraph();
}
```

5 Модель цифровой линии задержки

Составить программу, моделирующую линию задержки. Проиллюстрировать работу линии задержки с помощью финитного сигнала, поступающего на её вход. Число ячеек линии задержки выбрать произвольно, но не меньше числа отсчётов в финитном сигнале.

5.1 Иллюстрация работы одномерной линии задержки с выводом результатов в консоль

Вывести в консольное устройство содержимое одномерной линии задержки после каждого шага. Входной финитный сигнал выбирается согласно

варианту из табл. 1 п. 1.

Записать результат работы программы: значения отсчётов сигнала в линии задержки после каждого шага.

5.2 Графическая иллюстрация работы одномерной цифровой линии задержки

Вывести содержимое одномерной линии задержки в графическое окно. Входной финитный сигнал выбирается согласно варианту из табл. 1 п. 1. Зарисовать результат работы программы: отсчёты сигнала в линии задержки после каждого шага (см. п. 5.4).

5.3 Графическая иллюстрация работы двумерной цифровой линии задержки

Вывести содержимое двумерной линии задержки в графические окна. Отобразить синфазную линию X в верхнем окне, а квадратурную линию Y – в нижнем. Входной финитный квадратурный сигнал выбирается согласно варианту из табл. 2 п. 3 и табл. 1 п. 1. Зарисовать результат работы программы: отсчёты сигнала в линии задержки после каждого шага (см. п. 5.4).

5.4. Примечание к графическим программам

При выводе на экран содержимого линии задержки после каждого шага остановите вывод с помощью функции «`getch()`». В начале каждого очередного вывода содержимого линии задержки для каждого графического окна воспользуйтесь функцией «`.beginX(0)`», которая вызывается от объекта, представляющего собой графическое окно.

Пример

Моделирование одномерной цифровой линии задержки с помощью класса «Delay». Число ячеек линии задержки $N_{яч} = 7$. Цифровой сигнал, поступающий на вход линии задержки, задан в примере п. 1.

Модель одномерной линии задержки представлена в виде программы, приведённой ниже. Необходимо набрать программу, отладить, если допущены ошибки при наборе, и запустить на выполнение.

```
#include<iostream.h>
#include<conio.h>
#include<vect.h>
#include<delay.h>
void main()
{
int  i,LenDelay=7,T=5;
```

```
clrscr();
Delay<float> d(LenDelay);
vect<float> s(T);
for(i=0;i<s.len();i++)
 s[i]=(i+0.5)/T;
cout<<"Содержимое Л.З.: ";
d.print();
for(i=0;i<T+LenDelay;i++)
 {
 if(i<=s.up())   d.shift(s[i]);
 else            d.shift(0);
 cout<<"После "<<i+1<<"-го шага: ";
 d.print();
 }
getch();
}
```

6 Импульсная характеристика согласованного фильтра

Для варианта сигнала, заданного в п. 1, и вида модуляции, заданного в п. 3, вывести в графические окна компоненты элемента сигнала, а также значения весовых коэффициентов фильтра, согласованного с этим сигналом. Значения весовых коэффициентов совпадают со значениями отсчётов импульсной характеристики фильтра, согласованного с этим сигналом.

Импульсная характеристика (ИХ) согласованного фильтра (СФ) [3, §9.2]

$$\dot{g}(t) = a\dot{s}^{*}(T-t),$$

где постоянный коэффициент можно взять равным $a=1$, а «*» означает комплексное сопряжение.

6.1 Импульсная характеристика фильтра, согласованного с комплексным сигналом

Вывести отсчёты ИХ СФ в графические окна. Фильтр согласован с комплексным финитным сигналом. Комплексный финитный сигнал выбирается согласно варианту из табл. 1 п. 1. Зарисовать результат работы программы: отсчёты сигнала и отсчёты ИХ фильтра, согласованного с этим сигналом. Отобразить вещественную компоненту X в верхнем окне, а мнимую компоненту Y – в нижнем (см. п. 5.4).

6.2 Импульсная характеристика фильтра, согласованного с квадратурным сигналом

Вывести отсчёты ИХ СФ в графические окна. Фильтр согласован с квадратурным финитным сигналом. Квадратурный финитный сигнал выбирается согласно варианту из табл. 1 п. 1. Зарисовать результат работы программы: отсчёты сигнала и отсчёты ИХ фильтра, согласованного с этим сигналом.

Отобразить синфазную линию X в верхнем окне, а квадратурную линию Y – в нижнем (см. п. 5.4).

7 Модель согласованного фильтра

Для варианта сигнала, заданного в п. 1, и вида модуляции, заданного в п. 3 вывести в графические окна реакцию согласованного фильтра при приёме двоичных сигналов. Вероятности выдачи информационных сигналов взять из п. 4. Выделить моменты времени, когда на выходе согласованного фильтра обеспечивается максимум отношения пиковой мощности сигнала к средней мощности шума.

7.1 Фильтр, согласованный с комплексным сигналом

Вывести отсчёты сигнала и реакции СФ в графические окна. Фильтр согласован с комплексным финитным сигналом. Комплексный финитный сигнал выбирается согласно варианту из табл. 1 п. 1. Зарисовать результат работы программы: отсчёты сигнала и отсчёты реакции фильтра, согласованного с этим сигналом. Отобразить вещественную компоненту X в верхнем окне, а мнимую компоненту Y – в нижнем (см. пример п. 7).

7.2 Фильтр, согласованный с квадратурным сигналом

Вывести отсчёты ИХ СФ в графические окна. Фильтр согласован с квадратурным финитным сигналом. Квадратурный финитный сигнал выбирается согласно варианту из табл. 1 п. 1. Зарисовать результат работы программы: отсчёты сигнала и отсчёты реакции фильтра, согласованного с этим сигналом. Отобразить синфазную линию X в верхнем окне, а квадратурную линию Y – в нижнем (см. пример п. 7).

Пример

Исходный сигнал аналогичен сигналу из п. 3. Значения весовых коэффициентов совпадают со значениями отсчётов импульсной характеристики согласованного с сигналом фильтра. Значения отсчётов весовых коэффициентов располагаются в обратном порядке. Моменты времени, когда на выходе согласованного фильтра обеспечивается максимум отношения пиковой мощности к средней мощности шума, совпадают с моментами времени последнего отсчёта в каждом элементе сигнала.

```
#include<vect.h>
#include<quadr.h>
```

```
#include<oscgraph.h>
#include<filter.h>
void main()
{
int  j,i,bitX,bitY,mt=5;
quadr<float>  temp;
vect<quadr<float> >  s0(mt),s1(mt),g0(mt),g1(mt);
for(i=0;i<mt;i++)
 {
 s0[i].x()=(i+0.5)/mt;        s0[i].y()=1;
 s1[i].x()=-(i+0.5)/mt;       s1[i].y()=-1;
 g0[i].x()=1-(i+0.5)/mt;      g0[i].y()=1;
 g1[i].x()=-(1-(i+0.5)/mt);   g1[i].y()=-1;
 }
opengraph();
osc SX(1,  1,99,16), SY(1,51,99,66);
osc  F0X(1,17,99,32),  F0Y(1,67,99,82);
osc  F1X(1,33,99,48),  F1Y(1,83,99,98);
SX.rect(1);SX.assignMy(9);SX.step(2);
SY.rect(1);SY.assignMy(9);SY.step(2);
F0X.rect(1);F0X.assignMy(4);F0X.step(2);
F0Y.rect(1);F0Y.assignMy(4);F0Y.step(2);
F1X.rect(1);F1X.assignMy(4);F1X.step(2);
F1Y.rect(1);F1Y.assignMy(4);F1Y.step(2);
setcolor(12);
SX.PutMess("Компонента X сигнала на входе  СФ");
SY.PutMess("Компонента Y сигнала на входе  СФ");
F0X.PutMess("Компонента X сигнала на выходе СФ0");
F0Y.PutMess("Компонента Y сигнала на выходе СФ0");
F1X.PutMess("Компонента X сигнала на выходе СФ1");
F1Y.PutMess("Компонента Y сигнала на выходе СФ1");
filter<quadr<float>  >  F0(g0),F1(g1);
```

```
for(j=0;j<60;j++)
 {
 bitX=random(2);  bitY=random(2);
 for(i=0;i<mt;i++)
  {
  if(i==mt-1)
   SX.assignColor(13),SY.assignColor(13),
   F0X.assignColor(13),F0Y.assignColor(13),
   F1X.assignColor(13),F1Y.assignColor(13);
  else
   SX.assignColor(14),SY.assignColor(14),
   F0X.assignColor(14),F0Y.assignColor(14),
   F1X.assignColor(14),F1Y.assignColor(14);
  if(bitX)  temp.x()=s1[i].x();
  else      temp.x()=s0[i].x();
  if(bitY)  temp.y()=s1[i].y();
  else      temp.y()=s0[i].y();
  SX.dxl(temp.x());SY.dxl(temp.y());
  F0.shift(temp);
  F1.shift(temp);
  temp=F0.filtr();
  F0X.dxl(temp.x());F0Y.dxl(temp.y());
  temp=F1.filtr();
  F1X.dxl(temp.x());F1Y.dxl(temp.y());
  }
 }
getch(); closegraph();
}
```

8 Модель канала

Модель канала (рис. 2) включает в себя источник гауссовского шума. Шум моделируется функцией `noise(float sigma)` или `noise(compl<float> sigma)`. Здесь `sigma` – переменная, в которую записывается значение среднеквадратического отклонения для шума:

$$\sigma = \sqrt{\sigma^2},$$

где σ^2 – дисперсия шума. При моделировании гауссовского шума используется центральная предельная теорема. Число слагаемых в сумме равно 12. Формирование псевдослучайных слагаемых происходит с помощью функции `rand()` (см. приложение).

Для варианта сигнала, заданного в п. 1, и вида модуляции, заданного в п. 3 вывести в графические окна реакцию согласованного фильтра при приёме двоичных сигналов с аддитивным гауссовским шумом. Вероятности выдачи информационных сигналов взять из п. 4. Выделить моменты времени, когда на выходе согласованного фильтра обеспечивается максимум отношения пиковой мощности сигнала к средней мощности шума.

8.1 Фильтр, согласованный с комплексным сигналом

Вывести отсчёты сигнала и реакции СФ в графические окна. Фильтр согласован с комплексным финитным сигналом. Комплексный финитный сигнал выбирается согласно варианту из табл. 1 п. 1. Зарисовать результат работы программы: отсчёты сигнала и отсчёты реакции фильтра, согласованного с этим сигналом. Отобразить вещественную компоненту X в верхнем окне, а мнимую компоненту Y – в нижнем (см. пример п. 7).

8.2 Фильтр, согласованный с квадратурным сигналом

Вывести отсчёты ИХ СФ в графические окна. Фильтр согласован с квадратурным финитным сигналом. Квадратурный финитный сигнал выбирается согласно варианту из табл. 1 п. 1. Зарисовать результат работы программы: отсчёты сигнала и отсчёты реакции фильтра, согласованного с этим сигналом. Отобразить синфазную линию X в верхнем окне, а квадратурную линию Y – в нижнем (см. пример п. 7).

9 Модель для статистических испытаний

Для варианта сигнала, заданного в п. 1, и вида модуляции, заданного в п. 3, произвести статистический эксперимент.

А) Источник сигналов симметричный $P(0) = P(1) = 0{,}5$.

Б) Источник сигналов с заданными априорными вероятностями из п. 4.

Статистический эксперимент выполнить для отношений сигнал-шум, приведённых в табл. 5.

Таблица 5 – Значения параметра h^2 для статистических испытаний

h^2	0,5	1	2	3	4	5

Число испытаний: 10^5 (При отладке программы число испытаний взять 10^4).

Рассчитать вероятность ошибки при оптимальном приёме по критерию максимального правдоподобия [1, 2] для двоичных сигналов АМ, МОС(ЧМ) и ФМ.

Построить графики зависимости вероятности ошибки для АМ, МОС(ЧМ) и ФМ и частости ошибки (1), от отношения сигнал-шум по расчётным и экспериментальным результатам соответственно (индивидуально).

Вероятность ошибки для АМ, МОС (ЧМ) и ФМ:

$$p_{\text{ФМ}} = Q\left(\sqrt{2h^2}\right), \quad p_{\text{МОС или ЧМ}} = Q\left(\sqrt{h^2}\right), \quad p_{\text{АМ}} = Q\left(\sqrt{h^2/2}\right),$$

где $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} dt$ – функция ошибок.

На рис. 3 приведён пример построения графиков помехоустойчивости.

Рис. 3 – Пример построения кривых помехоустойчивости и доверительных интервалов

На рис. 3 ось абсцисс – линейный масштаб для отношения сигнал-шум $p_{\text{ош}}$, а ось ординат – логарифмический масштаб для вероятности (частости) ошибки h^2.

Сопоставить теоретические и экспериментальные результаты. Для этого необходимо построить доверительные интервалы.

Доверительные интервалы находятся по следующим формулам

$$p_1 = p - \Phi^{-1}\left(P_{\text{дов}}\right)\sqrt{\frac{p(1-p)}{n}}, \qquad p_2 = p + \Phi^{-1}\left(P_{\text{дов}}\right)\sqrt{\frac{p(1-p)}{n}},$$

где доверительная вероятность $P_{\text{дов}} = 0{,}95$ или $P_{\text{дов}} = 0{,}99$, число испытаний: n = 100000, p – рассчитанная вероятность ошибки, $\Phi^{-1}(x)$ – функция, обратная функции Крампа

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-x}^{x} \mathrm{e}^{-t^2/2} dt .$$

Сделать вывод о соответствии результатов моделирования теоретическим расчётам.

Для нахождения значений функций $Q(x)$ и $\Phi(x)$ воспользуйтесь справочником [4, стр. 76–77], где табулированы значения $\Phi_0(x)$. При этом

$$\Phi_0(x) = \frac{1}{\sqrt{2\pi}} \int_{0}^{x} e^{-t^2/2} dt = \frac{1}{2}\Phi(x) = \frac{1}{2} - Q(x) .$$

Пример

Ниже приведён пример программы для сигнала, приведённого в примере для п. 3. В этом примере используется равновероятный источник сигналов. Для источника сигнала с заданными априорными вероятностями появления символов программу необходимо изменить. При этом необходимо рассчитать порог принятия решения демодулятором и учесть это при коррекции программы.

```
#include<vect.h>
#include<noise.h>
#include<conio.h>
#include<filter.h>
#include<compl.h>
void main()
{
long err,k,N;
int i,j,bit,r,mt;
int q; //Задать вид модуляции: 0 - АМ, 1 - ФМ, 2 - МОС или ЧМ
float a,h2;
float E; // энергия ненулевой посылки
randomize();
cout<<"Введите число испытаний"<<endl;
cin>>N;
cout<<"Число микротактов"<<endl;
cin>>mt;
compl<float> z,sigma,ns;
```

```
vect<compl<float> > s0(mt),s1(mt); // массив сигналов
vect<compl<float> > g0(mt),g1(mt); // массив импульсных
                  //характеристик согласованных фильтров
float temp;
for(;;)
 {
 cout<<"Введите отношение сигнал-шум"<<endl;
 cout<<" (Для завершения работы введите -1)"<<endl;
 cin>>h2;
 if(h2<0) break;
 // Задать сигнал
 // Найти энергию ненулевой посылки
 // Найти импульсные характеристики согласованных фильтров
 filter<compl<float> > F0(g0),F1(g1);
 if(h2)
  sigma(sqrt(E/2/h2),sqrt(E/2/h2));
 else
  sigma(1e10,1e10);
 cout<<"sigma="<<sigma<<endl;
 for(err=k=0;k<N;k++)
  {
  bit=random(2); // здесь заданы равновероятные символы
  if(bit)
   for(i=0;i<mt;i++)
    {
    ns=noise(sigma);
    z=s1[i]+ns;
    F0.shift(z);
    F1.shift(z);
    }
  else
   for(i=0;i<mt;i++)
```

```
    {
    ns=noise(sigma);
    z=s0[i]+ns;
    F0.shift(z);
    F1.shift(z);
    }
  switch(q)
   {
   case 0: // Для АМ
     r=F0.filtr().x()>E/2 ? 0 : 1;
    break;
   case 1: // Для ФМ
     r=F0.filtr().x()>0 ? 0 : 1;
    break; // Для МОС и ЧМ
   case 2:
     r=F0.filtr().x()>F1.filtr().x() ? 0 : 1;
    break;
   default:
     cerr<<"Не задан вид модуляции"<<endl;
    goto end;
   }
  err+=r^bit;
  }
 cout<<"h^2="<<h2<<"\tNerr="<<err<<endl;
 }
end:
cout<<"Моделирование завершено"<<endl;
}
```

ПРИЛОЖЕНИЯ

П.1 Описание класса `osc` (из файла `osc.h`)

Этот класс предназначен для отображения как дискретных, так и непрерывных сигналов во времени (имитация осциллографа). При создании экземпляра класса (объекта) необходимо задать координаты окна, в котором будут отображаться сигналы. Для данного класса координаты (x, y) на экране монитора задаются вещественными числами. Точка (0, 0) соответствует левой верхней точке на экране.

Точка (100, 100) соответствует правой нижней точке на экране.

Конструктор объекта, принадлежащего классу `osc`:

```
osc(float x1, float y1, float x2, float y2)
```

Координаты x_1 и y_1 соответствуют верхнему левому углу окна.

Координаты x_2 и y_2 соответствуют нижнему правому углу окна.

На рис. 4 показано расположение окна, созданное экземпляром класса `osc` на экране монитора.

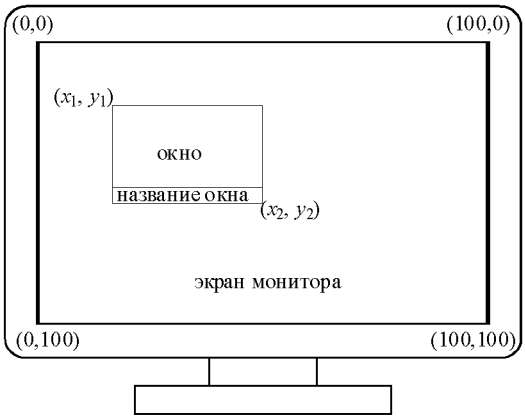

Рис. 4 – Отображение окна, созданное объектом класса osc на экране монитора

Пример создания объекта

```
osc window(12.2,16.6,65.1,65.0);
```

`osc` – тип объекта; `window` – название объекта;

`12.2,16.6,65.1,65.0` – координаты окна x_1, y_1, x_2, y_2, созданного объектом.

П.2 Функции класса osc

`rect(int color);` – рисует границы окна цветом `color`.

`assignMy(int msh);` – устанавливает масштаб по оси ординат, где `msh` – число пикселей, отображающих значение, равное 1.

`dx(float y);` – рисование непрерывной линии: соединяет точки предыдущего значения с текущим значением *y* по оси ординат, а по оси абсцисс происходит смещение на 1 пиксель.

`dxl(float y);` – рисование отсчётов сигнала: вертикальные линии от оси до текущего значения *y* по оси ординат.

`step(int st);` – установка шага в пикселях `st`: влияет на рисование отсчётов с помощью функции `dxl`, при этом отсчёты сигнала отображаются через `st` пикселей по оси абсцисс.

`PutMess(char* s);` – выдаётся сообщение (символьная строка `s`) в окне, пример на рисунке "название окна".

Примеры использования некоторых функций

`window.rect(1);` – контуры окна рисуются тёмно-синим цветом, при этом форма окна на экране показана на рис. 4.

`window.PutMess("Название окна");` – комментарий в нижней части окна, пример отображения комментария показан на рис. 4.

П.3 Описание некоторых других функций

`opengraph();` – открыть графический режим VGA (640×480 пикселей);

`closegraph();` – закрыть графический режим и перейти к символьному режиму (из файла `graphics.h`);

`color(int c);` – установить цвет рисования (для VGA режима `c=0...15`, функция из файла `graphics.h`);

`getch();` – ожидание ввода с клавиатуры (из файла `conio.h`);

`clrscr();` – очистка символьного экрана (из файла `conio.h`);

`random(int n);` – функция, выдающая целые псевдослучайные числа равномерно распределённые в диапазоне от `0` до `n-1` (из файла `stdlib.h`).

П.4 Некоторые функции и константы, определённые в математической библиотеке `math.h`

`double sin(double x);` – синус;

`double exp(double x);` – экспонента;

`double pow(double x, double y);` – возведение `x` в степень `y`;

`double sqrt(double x);` – квадратный корень;

`M_PI` – Константа π;

M_PI_2 – Константа π/2.

double fabs(double x); – модуль.

Пример использования функций

```
y=sin(exp(-pow(x,3)))/sqrt(2*M_PI);
```

П.5 Описание некоторых функций класса комплексных чисел из библиотечного файла compl.h и квадратурных чисел из библиотечного файла quadr.h

compl<TYPE>(TYPE x, TYPE y);

– конструктор класса compl;

quadr<TYPE>(TYPE x, TYPE y);

– конструктор класса quadr (наследует свойства класса compl);

compl<float> z;

– объявление комплексной переменной, где вещественная и мнимая части представляют собой числа формата float;

quadr<double> z;

– объявление квадратурной переменной, где X и Y компоненты представляют собой числа формата double;

TYPE& compl<TYPE>::x(); или compl<TYPE>::X;

– получить доступ на чтение и запись к вещественной части комплексного числа;

TYPE& compl<TYPE>::y(); или compl<TYPE>::Y;

– получить доступ на чтение и запись к мнимой части комплексного числа;

TYPE& quadr<TYPE>::x(); или quadr<TYPE>::X;

– получить доступ на чтение и запись к X компоненте квадратурного числа;

TYPE& quadr<TYPE>::y(); или quadr<TYPE>::Y;

– получить доступ на чтение и запись к Y компоненте квадратурного числа;

TYPE& quadr<TYPE>::y(); или quadr<TYPE>::Y;

compl<TYPE> conj(); – комплексное сопряжение;

TYPE mod2(); – квадрат модуля комплексного числа.

Примеры

```
compl<float> z(1.0,-2.5);
```

– объявление комплексной переменной, у которой вещественная и мнимая

части представляют собой числа формата `float`, и инициализация вещественной части числом `1.0`, а мнимой части числом `-2.5`;

```
z.x();
```

– получить доступ к вещественной части комплексного числа `z`;

```
quadr<int> q(10,-5);
```

– объявление квадратурной переменной, где синфазная и квадратурная части представляют собой числа формата `float`, и инициализация синфазной части числом `1.0`, а квадратурной части числом `-2.5`;

```
cout<<z.y()<<endl;
```

– вывести на экран значение мнимой части комплексного числа `z`;

```
z.x()=10.4;
```

– присвоить вещественной части комплексного числа `z` значение, равное `10.4`;

```
q.x()=-0.5;
```

– присвоить X компоненте квадратурного числа `q` значение, равное `-0.5`;

`q.Y=7;` – присвоить Y компоненте квадратурного числа `q` значение, равное 7.

П.6 Описание класса одномерного динамического массива `vect` (из файла `vect.h`)

Подробное описание класса `vect.h` приведено в [6, § 6.5, § 9.4]. Конструктор класса:

```
vect<T>(int n);
```

где `n` – размер одномерного динамического массива;
`T` – имя типа в шаблонной подстановке;

```
T& vect<T>::operator[](int i);
```

– оператор «квадратные скобки», реализующий доступ на запись и чтение содержимого ячеек одномерного динамического массива;

```
int vect<T>::len();
```

– функция, возвращающая размер одномерного динамического массива;

```
int vect<T>::up();
```

– функция, возвращающая индекс последнего элемента в одномерном динамическом массиве.

Примеры

```
vect<long double> mas(2);
```

– объявить одномерный динамический массив с именем mas, содержащий 2 элемента, каждый элемент массива имеет тип long double, все элементы массива инициализируются нулевым значением;

```
mas[1]=7.89;
```

```
cout<<"Массив, размером в "<<mas.up()
```

```
<<" элемента, имеет значение последнего\
```

```
элемента, равное "<<mas[mas.up()]<<endl;
```

– выводится следующее сообщение: «Массив, размером в 2 элемента имеет значение последнего элемента, равное 7,89».

П.7 Описание класса Delay (из файла delay.h)

Конструктор класса:

```
Delay<T>(int n);
```

где n – число ячеек в линии задержки (длина линии задержки);
T – имя типа в шаблонной подстановке;

```
T Delay<T>::shift(T z);
```

– функция, реализующая сдвиг в линии задержки и запись отсчёта сигнала z во входную ячейку линии задержки, результатом функции является выталкиваемое значение, которое находилось в последней ячейке линии задержки;

```
T Delay<T>::operator[](int i);
```

– оператор «квадратные скобки», реализующий доступ на чтение содержимого элемента линии задержки с номером i, входная ячейка линии задержки имеет индекс, равный 0;

```
void Delay<T>::len();
```

– функция, возвращающая размер линии задержки;

```
void Delay<T>::print();
```

– функция вывода на экран в символьном виде содержимого всей линии задержки.

Примеры

```
Delay<long double> sig(10);
```
– создание линии задержки с именем sig размером в 10 ячеек, каждая ячейка имеет тип long double;

```
length=7;
```

```
Delay<quadr<double> > complDelay(length);
```

– создание линии задержки с именем complDelay размером в length ячеек (в данном примере, т.к. length=7, то и размер линии задержки равен 7), каждая ячейка имеет тип quadr<double>;

```
long double z=10.1;
```

```
sig.shift(z)
```

– запись отсчёта z, равного 10.1, в линию задержки с именем sig;

```
cout<<sig[0]<<'\t'<<sig[sig.len()-1]<<endl;
```

– вывод на экран в символьном виде содержимого первой (входной) и последней ячеек линии задержки с именем sig;

```
sig.print();
```

– вывод на экран в символьном виде содержимого всей линии задержки с именем sig;

П.8 Описание класса filter (из файла filter.h)

Конструктор класса цифрового фильтра filter (наследует свойства класса линии задержки Delay):

```
filter<T>(vect<T> ir);
```

где

```
vect<T> ir
```

– одномерный динамический массив с шаблонной (template) подстановкой, содержащий отсчёты импульсной характеристики цифрового фильтра; T – имя типа в шаблонной подстановке;

```
T filter<T>::shift(T z);
```

– функция (результат наследования), реализующая сдвиг в линии задержки цифрового фильтра и запись отсчёта сигнала z во входную ячейку линии задержки цифрового фильтра, результатом функции является выталкиваемое значение, которое находилось в последней ячейке линии задержки цифрового фильтра;

```
T filter<T>::operator[](int i);
```

– оператор «квадратные скобки» (результат наследования), реализующий доступ на чтение содержимого элемента линии задержки цифрового фильтра с номером (индексом) i, входная ячейка линии задержки цифрового фильтра имеет индекс, равный 0;

```
T& filter<T>::operator()(int i);
```

– оператор «круглые скобки», реализующий доступ на запись и чтение содержимого весовых коэффициентов линии задержки цифрового фильтра с номером (индексом) i, направление возрастания индексов весовых коэффициентов совпадает с направлением возрастания индексов ячеек линии задержки цифрового фильтра;

```
int filter<T>::len();
```

– функция (результат наследования), возвращающая размер линии задержки цифрового фильтра;

```
T filter<T>::filtr();
```

– функция, возвращающая результат фильтрации цифрового фильтра;

```
void filter<T>::print();
```

– функция вывода на экран в символьном виде содержимого линии задержки и весовых коэффициентов цифрового фильтра.

Примеры

```
vect<int> ir(2)
ir[0]=1, ir[1]=-1,
filter<int> dk(ir);
```

– создание цифрового фильтра с именем dk (2 ячейки линии задержки, каждая ячейка имеет тип int и 2 весовых коэффициента, также имеющих тип int;

```
int z=10;
dk.shift(z);
```

– запись отсчёта z, равного 10, в цифровой фильтр с именем dk;

```
cout<<dk.filtr()<<endl;
```

– вывести на экран результат фильтрации цифрового фильтра с именем dk;

```
cout<<dk[0]<<'\t'<<dk(0)<<endl;
```

– вывод на экран в символьном виде содержимого первой (входной) ячейки линии задержки и значения первого весового коэффициента цифрового фильтра с именем dk;

```
dk.print();
```

– вывод на экран в символьном виде содержимого линии задержки и значения весовых коэффициентов цифрового фильтра с именем dk;

П.9 Описание функции noise (из файла noise.h)

```
TYPE noise(TYPE sigma);
```

– возвращает случайное число, имеющее гауссовское распределение, на входе параметр среднеквадратического отклонения (СКО): $\sigma = \sqrt{\sigma^2} = \sqrt{D}$, где D – значение дисперсии гауссовского распределения.

Пример

```
cout<<noise(2)<<endl;
```

– возвращает случайное число, имеющее гауссовское распределение с дисперсией, равной 4.

ЛИТЕРАТУРА

1. Теория электрической связи: Учебник для вузов / А. Г. Зюко, Д. Д. Кловский, В. И. Коржик, М. В. Назаров; под ред. Д. Д. Кловского. – М.: Радио и связь, 1998. – 430 с.

2. Прокис Дж. Цифровая связь. Пер с англ. / Под ред. Д. Д. Кловского. – М.: Радио и связь. 2000. – 800 с.

3. Ван Трис Г. Теория обнаружения, оценок и модуляции. Том III. Обработка сигналов в радио- и гидролокации и приём случайных гауссовых сигналов на фоне помех. Нью-Йорк, 1971. Пер. с англ. Под ред. проф. В.Т. Горяинова. М., «Сов. Радио», 1977. – 664 с.

4. Справочник по математике для инженеров и учащихся втузов. Бронштейн И. Н., Семендяев К.А. – М. Наука. 1981. – 713 с.

5. Язык С. Руководство для начинающих. Уэйт М., Прата С., Мартин Д., М., Мир 1988. – 512 с.

6. Объектно-ориентированное программирование на С++. А. Пол СПб., М., «Невский диалект», «Изд. БИНОМ» 1999. – 462 с.

СОДЕРЖАНИЕ