Комментарий для препода: Альфа версия, нужно заменить график с режимами тестированием и переписать кое-какие лоскутки – можно (нужно) проверять!

Последний этап– это тестирование декодирующих способностей кодека. Во-первых, необходимо удостовериться в том, что разработанный кодек кодирует и декодирует информацию так же качественно, как и исходный. Во-вторых, необходимо сравнить декодирующие способности всех трех режимов кодирования. Для того, чтобы выполнить поставленные задачи потребовалось написать отдельную программу, которая бы запускала модули кодека в необходимом порядке, передавая на кодирование различные данные, сгенерированные псевдослучайным образом. Алгоритм разработанной программы показан на рисунке 1.

Рис. 1. Алгоритм тестирования кодека

Работа программы происходит следующим образом:

В цикле генерируется случайная последовательность нулей и единиц. Затем сгенерированная последовательность передается на вход кодера, где она подвергается кодированию и чередованию. После кодирования данные готовы для отправки в канал. Так как были поставлены сразу две задачи (тестирование работоспособности и проведение сравнительного анализа различных режимов), модуль имитирующий шум в канале, в первом случае запускаться не будет (об этом модуле ниже). После получения данных из «канала», происходит ликвидация чередования и декодирование данных. По завершению декодирования полученная и искомая последовательности сравниваются. В случае если они не совпали, номер теста и последовательность записываются в базу данных.

Результатом тестирования стало обнаружение трех ошибок: одной в основном теле программы и двух в добавленном режиме. После исправления допущенных недочетов, тестирование было повторено, и ошибок не выявило.

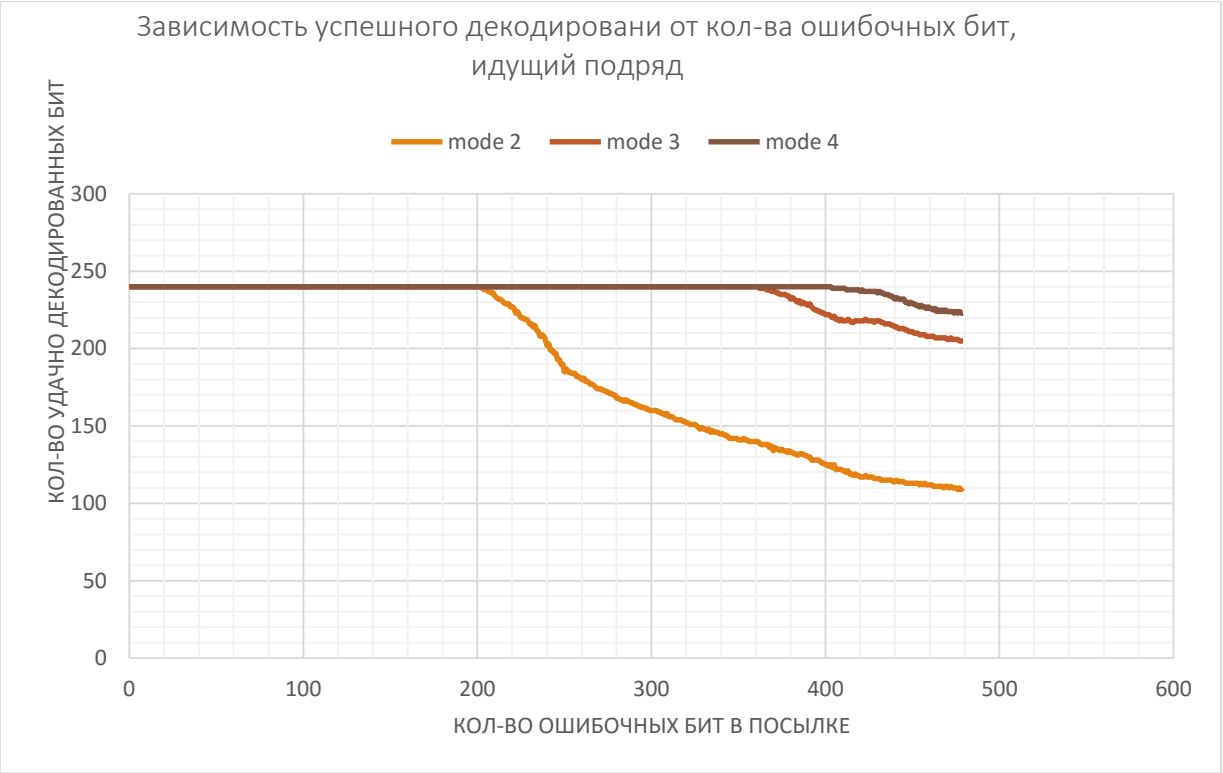
Сразу после того, как тесты подтвердили работоспособность кодека в «холостом режиме», без зашумления данных в канале, необходимо проверить и подтвердить восстанавливающие способности разработанного декодера. Для этого в тело цикла тестирования включается модуль имитирующий шум в канале.

Данный модуль имеет три входных параметра, задаваемых перед запуском модуля: количество зашумлённых участков в передаваемой посылке, длина первого зашумленного участка (если участков n, каждый следующий будет на (100/n) % короче первого) и расстояние между зашумленными участками (в случае, если данный параметр не указан – зашумленные участки создаются на максимальном расстоянии друг от друга, имитируя наихудший вариант. Это объясняется работой модуля чередования, т.к. он распределяет данные равномерно на максимальном расстоянии друг от друга). Во избежание получения однотипных результатов, модуль содержит ряд случайно сгенерированных внутренних параметров (например, расстояние, от начала посылки, на котором генерируется первое зашумление). Биты, которые необходимо подвергнуть зашумлению – инвертируются и им присваивается псевдослучайный вес.

Для тестирования восстанавливающих способностей декодера были проведены эксперименты со следующими настройками модуля-шума:

№	Кол-во ошибочных последовательностей (ОП)	Кол-во испорченных бит	Расстояние между ОП
1	1	1-492	-
2	2	1-360; 1-180;	0-120
3	3	1-240; 1-120; 1-60;	0-120

В результате проведенных опытов зависимость от количества ОП не выявилась. Все опыты оказались схожи по результату. Поэтому на диаграмме 1 предоставлен только один опыт (первый).



На диаграмме 1 предоставлена визуальная характеристика декодирующих способностей различных режимов кодирования. Эксперимент был поставлен следующим образом: генерируется порядка тысячи случайных последовательностей по 240 бит каждая. Последовательности кодируются, затем в каждую из закодированных посылок вносится N ошибок (ось абсцисс), после чего сообщения декодируются. По завершению декодирования все последовательности сравниваются с исходными. В случае, если исходная и декодированная последовательность не совпали – подсчитывает кол-во несовпадений. Затем берётся среднее арифметическое всех ошибок (с округлением в большую сторону) и отнимается от исходной длины кодируемой последовательности (от 240). То есть, если на 1000 случайных сообщений все ошибки, возникшие в канале удачно было исправлены, на графике по оси ординат будет 240. В случае, если была 1,2 или более ошибок, прямая искривляется. Нас интересует только удачно декодированные последовательности (именно поэтому считали ср. арф.).

Из результатов видно следующее:

Режим 2 способен исправить до 201 ошибочных бит из 492 бит передаваемого сообщение (240 * 2 + 12 бит для декодера Витерби).

Режим 3, добавленный в ходе разработки, способен удачно декодировать сообщение при 360 ошибках из 738 передаваемых бит (240*3+18 служебных бит для декодера).

Режим 4, удачно декодирует сообщение при немного большем - 400 ошибках из 984 передаваемых бит (240*4+6*4 служебных бит для декодера).

Может показаться, что разработанный режим кодирования превосходит оба режима по надежности (успешное декодирование происходит в среднем при ошибках до 49% от длины посылки, в то время как у режима 2 и 4 – это значение равно 41%), но это не так. Дело в том, что каждый из режимов создан для канала с соответствующими свойствами. Если в канале возникают ошибки длиной до 150-200 бит – используют режим кодирования 2, если ошибки до 400 бит, то режим 4, но при этом скорость падает в 2раза, относительно режима 2. В случае же, если использовать разработанный режим 3 – длина последовательности ошибок в канале не должна

превышать 49% от всей длины сообщения, но при этом скорость падает не в 2 раза, а в 1,5 (738 против 984 передаваемых бит, при кодировании 240 исходных бит).